creative commons
COMMONS DEED

저작자표시-비영리-변경금지 2.0 대한민국

이용자는 아래의 조건을 따르는 경우에 한하여 자유롭게

- 이 저작물을 복제, 배포, 전송, 전시, 공연 및 방송할 수 있습니다.

다음과 같은 조건을 따라야 합니다:

저작자표시. 귀하는 원저작자를 표시하여야 합니다.

비영리. 귀하는 이 저작물을 영리 목적으로 이용할 수 없습니다.

변경금지. 귀하는 이 저작물을 개작, 변형 또는 가공할 수 없습니다.

- 귀하는, 이 저작물의 재이용이나 배포의 경우, 이 저작물에 적용된 이용허락조건을 명확하게 나타내어야 합니다.
- 저작권자로부터 별도의 허가를 받으면 이러한 조건들은 적용되지 않습니다.

저작권법에 따른 이용자의 권리는 위의 내용에 의하여 영향을 받지 않습니다.

이것은 이용허락규약(Legal Code)을 이해하기 쉽게 요약한 것입니다.

Disclaimer

dCollection

이학 박사 학위논문

A numerical study on level set based multiphase flow simulation

(레벨셋 방법을 이용한 다층 유체 시뮬레이션의 수치적 연구)

2015년 8월

서울대학교 대학원

수리과학부

이병준

A numerical study on level set based multiphase flow simulation

A dissertation
submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
to the faculty of the Graduate School of
Seoul National University

by

Byungjoon Lee

Dissertation Director : Professor Myungjoo Kang

Department of Mathematical Sciences
Seoul National University

August 2015

© 2015 Byungjoon Lee

All rights reserved.

Abstract

This thesis concerns numerical methods for simulating multiphase flow using level set method. The motion of multiphase flow can be expressed as the incompressible Navier-Stokes equation. First, we survey numerical methods that can approximately solve the governing equations. Also, we introduce level set method for describing interface of fluid and show how to combine level set method with the Navier-Stokes equations. Finally, we show numerical simulation by core-annular flow in horizontal pipe problem.

Key words: Multiphase flow, The Navier-Stokes equation, Level Set method, Core-Annular Flow, Numerical Simulation
Student Number: 2011-30900

Contents

List of Figures

LIST OF FIGURES

Chapter 1

Introduction

The multiphase flow simulation is important for many industrial application, including computer graphics, computational physics and engineering. In spite of numerous needs on this simulation, it is still difficult to achieve accuracy and efficiency simultaneously. When we attempt to describe the motion of multiphase flow, two things must be considered: To obtain the information of properties of fluid itself, such as velocity of fluid, and to express motion of interface.

Mathematical answer for former one is to solve the incompressible Navier-Stokes Equations. For many years, tons of numerical methods have been developed for solving the incompressible Navier-Stokes Equations for multiphase flow [3, 4, 6, 8, 11, 12, 13, 21, 22]. In this thesis, we focus on the way in [3], so called, the projection method. The reason behind we choose this method is as follows: Finite Difference Method based method and easy to combine for Navier-Stokes solver with level set method.

Multiphase incompressible flow algorithms for describing interface of multiphase flow problems include vortex method [12], boundary integral method [13], volume of fluid method [14], front tracking methods [11]. As we have seen in [11], numerous diffusion which destroy the sharpness of the front can be incurred in conventional conservative schemes. High order conservative schemes produce numerical oscillations around the front. Also, since the conventional algorithms for describing the motion of fluid is tracking the front, those have difficulties to implement, that is, hard to implement. These prob-

lems are amplified when solving a three dimensional problem.
In [6], a level set formulation for moving interface has been developed. The level set function is a smooth function, denoted as ϕ, which eliminates the problems that conventional difference schemes incur. Furthermore, the level set formulation generalizes to three dimensions, well. The actual front location never has to be computed. Instead, the front is embedded as a particular level set in a fixed domain PDE.
In this thesis, mathematical formulation for multiphase incompressible flow motion including governing equation and level set method is given in Chapter 2. Various numerical methods for solving the incompressible Navier-Stokes equations and level set advection are given in Chapter 3. Finally, in Chapter 4, computational results of multiphase flow simulation is illustrated by core-annular flow example.

Chapter 2

Mathematical Formulation of Fluid Motion

In this chapter, we introduce governing equations of fluid motion, which is Navier-Stokes equations.

2.1 Governing Equations

2.1.1 The Equation of Motion - Navier-Stokes Equations

The motion of multiphase incompressible flow can be expressed with the Navier-Stokes equation:

$$u_t + \mathbf{u} \cdot \nabla u + \frac{p_x}{\rho} = \frac{(2\mu u_x)_x + (\mu(u_y + v_x))_y + (\mu(u_z + w_x))_z}{\rho} \tag{2.1}$$

$$v_t + \mathbf{u} \cdot \nabla v + \frac{p_y}{\rho} = \frac{(\mu(u_y + v_x))_x + (2\mu v_y)_y + (\mu(v_z + w_y))_z}{\rho} + g \tag{2.2}$$

$$w_t + \mathbf{u} \cdot \nabla w + \frac{p_z}{\rho} = \frac{(\mu(u_z + w_x))_x + (\mu(v_z + w_y))_y + (2\mu w_z))_z}{\rho} \tag{2.3}$$

By rewriting above equations into vector form, we have

$$\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} = \frac{1}{\rho}(-\nabla p + \nabla \cdot (2\mu S)) + \mathbf{F} \tag{2.4}$$

where ρ is the density, μ the viscosity, and S the viscous stress tensor,

$$S_{ij} = \frac{1}{2}(\frac{\partial u_j}{\partial x_i} + \frac{\partial u_i}{\partial x_j}) \tag{2.5}$$

and $\mathbf{F}$ is the source term, such as acceleration exerted by gravity and surface tension, and p is a pressure. Here, $\mathbf{F}$ is considered only to be gravity acceleration and surface tension effect will be added later discussion.
From incompressibility condition of the given flow, the velocity $\mathbf{u}$ must be subject to the following condition:

$$\nabla \cdot \mathbf{u} = 0 \tag{2.6}$$

2.1.2 Dimensionless Form of Navier-Stokes Equations

In order to compare properties of flows, the dimensionless form of Eq.(2.4) can be used in general. The following variables are used for expressing dimensionless form:

$$\begin{array}{lll} \mathbf{x} = Rx^*, & \mathbf{u} = V_0 u^*, & t = \frac{R}{V_0} t^* \\ p = p^* \rho_0 (V_0)^2, & \rho = \rho_0 \rho^*, & \mu = \mu_0 \mu^* \end{array} \tag{2.7}$$

where dimensionless variables are denoted as $*$; R is the reference length; and V_0 is the reference velocity, ρ_0 the reference density, μ_0 the reference viscosity. Transforming Eq.(2.1) to (2.3) into dimensionless form and dropping $*$, we have

$$u_t + \mathbf{u} \cdot \nabla u + \frac{p_x}{\rho} = \frac{1}{Re} \frac{(2\mu u_x)_x + (\mu(u_y + v_x))_y + (\mu(u_z + w_x))_z}{\rho} \tag{2.8}$$

$$v_t + \mathbf{u} \cdot \nabla v + \frac{p_y}{\rho} = \frac{1}{Re} \frac{(\mu(u_y + v_x))_x + (2\mu v_y)_y + (\mu(v_z + w_y))_z}{\rho} + \frac{R}{V_0^2} g \tag{2.9}$$

$$w_t + \mathbf{u} \cdot \nabla w + \frac{p_z}{\rho} = \frac{1}{Re} \frac{(\mu(u_z + w_x))_x + (\mu(v_z + w_y))_y + (2\mu w_z)_z}{\rho} 6 \tag{2.10}$$

where the Reynolds number, $Re = \frac{\rho_0 R V_0}{\mu_0}$

2.2 Level Set Method

Besides velocity information obtained from Navier-Stokes Equations, we need a tool for expressing the flow interface for describing motion of fluid. The level set technique is used to describe the interface [18]. We initialize ϕ_f to be the signed distance function from the interface between two phases. Then, the interface can be expressed as the zero level set of ϕ_f as

$$\Gamma = \{\mathbf{x} \mid \phi_f(\mathbf{x}, t) = 0\} \tag{2.11}$$

The fluid 1 and fluid 2 regions will be described by $\phi_f < 0$ and $\phi_f > 0$. Then, velocities in each phase can be written as

$$\mathbf{u} = \begin{cases} u_1, & \phi_f \leq 0 \\ u_2, & \phi_f > 0. \end{cases} \tag{2.12}$$

where u_i,$i = 1, 2$, is the velocity of each flow. When using the level set method, the level set function is set to move along with the given velocity through the following equation:

$$(\phi_f)_t + \mathbf{u} \cdot \nabla \phi_f = 0. \tag{2.13}$$

This level set method has many advantages on capturing interface. First of all, the level set function ϕ_f always remains a function if speed function is smooth. This guarantees that level set method can well capture topological changes such as break, merge. And also, the level set approach can be easily extended to higher dimension. This property allows us to express phenomena of fluid dynamics in real world, which is three dimensional space. Finally, intrinsic geometric properties of the interface can be easily calculated from the level set function ϕ_f. For example, the unit normal on the interface and curvature of the interface can be computed as

$$\mathbf{n} = \frac{\nabla \phi_f}{|\nabla \phi_f|} \tag{2.14}$$

$$\kappa = \nabla \cdot \mathbf{n} \tag{2.15}$$

where the unit normal is drawn from the fluid 1 into the fluid 2.
For simplicity, we use ϕ instead of the flow level set function ϕ_f for the rest of this thesis.

Chapter 3

Numerical Methods

This chapter will give numerical technique for solving Navier-Stokes equation and level set evolution. For simplicity, we denote each part of the following Navier-Stokes equations as

$$\boxed{\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u}}_{Advection} = \boxed{-\frac{1}{\rho}\nabla p}_{Projection} + \boxed{\frac{1}{\rho}\nabla \cdot (2\mu S)}_{Viscosity} + \mathbf{F} \quad (3.1)$$

$$\boxed{\phi_t + \mathbf{u} \cdot \nabla \phi}_{Advection} = 0. \quad (3.2)$$

Note that the name of each term originated from physical phenomena.

3.1 The Projection Method

This method was initially proposed by Chorin [3], whereas an explicit version of the method was presented by Fortin et al.(1971). This explicit method is a fractional step method with first-order accuracy in time. Let $\mathbf{u}^n = (u^n, v^n, w^n)$. In the first step, we explicitly compute a provisional value, u^*, with

$$\frac{\mathbf{u}^* - \mathbf{u}^n}{\triangle t} + (\mathbf{u}^n \cdot \nabla)\mathbf{u}^n - \frac{1}{Re}\nabla \cdot (2\mu S) - \mathbf{F} = 0 \quad (3.3)$$

which is the momentum equation with no pressure gradient. Only the discretization in time is considered here. Then, in the second step, we correct

$\mathbf{u}^*$ by considering the equations

$$\frac{\mathbf{u}^{n+1} - \mathbf{u}^*}{\triangle t} + \frac{1}{\rho}\nabla p^{n+1} = 0 \tag{3.4}$$

$$\nabla \cdot \mathbf{u}^{n+1} = 0 \tag{3.5}$$

By using the divergence of Eq.(3.4) and by making use of Eq.(3.5) which states that $\mathbf{u}^{n+1}$ must be a divergence-free vector, we gain the Poisson equation:

$$\nabla \cdot (\frac{1}{\rho}\nabla p^{n+1}) = \frac{1}{\triangle t}\nabla \cdot \mathbf{u}^* \tag{3.6}$$

The boundary condition for p is obtained by projecting vector equation (3.4) on the outward normal unit $\mathbf{n}$ to the boundary Γ. Thus, we obtain the Neumann condition:

$$(\frac{\partial p^{n+1}}{\partial \mathbf{n}})_\Gamma = -\frac{\rho}{\triangle t}(\mathbf{u}_\Gamma^{n+1} - \mathbf{u}_\Gamma^*) \cdot \mathbf{n} \tag{3.7}$$

where $\mathbf{u}_\Gamma^*$ is the value of $\mathbf{u}^*$ on Γ. To summarize, Eq.(3.3) produces u^*; the solution of the Neumann problem (Eq.(3.6)) produces p^{n+1}; finally, the velocity, $\mathbf{u}^{n+1}$, is computed from Eq.(3.4). More details are provided in a previous study [4].

3.2 Advection Term

The spatial advection terms in Eqs.(3.1) and (3.2) can be discretized as follows:

$$((\mathbf{u}\cdot\nabla)u)_{i+1/2,j,k} = \frac{u_{i+1/2,j,k}(u_{i+1,j,k}-u_{i,j,k})}{\triangle x} + \frac{v_{i+1/2,j,k}(u_{i+1/2,j+1/2,k}-u_{i+1/2,j-1/2,k})}{\triangle y} + \frac{w_{i+1/2,j,k}(u_{i+1/2,j,k+1/2}-u_{i+1/2,j,k-1/2})}{\triangle z} \tag{3.8}$$

$$((\mathbf{u}\cdot\nabla)\phi)_{i,j,k} = \frac{u_{i,j,k}(\phi_{i+1/2,j,k}-\phi_{i-1/2,j,k})}{\triangle x} + \frac{v_{i,j,k}(\phi_{i,j+1/2,k}-\phi_{i,j-1/2,k})}{\triangle y} + \frac{w_{i,j,k}(\phi_{i,j,k+1/2}-\phi_{i,j,k-1/2})}{\triangle z} \tag{3.9}$$

Velocity advection for v and w are treated similarly. In Eqs.(3.8) and (3.9), $u_{i+1,j,k}$, $\phi_{i+1/2,j,k}$ must be obtained by using proper approximation. For more explanation, we will consider the following general advection equation:

$$u_t + f(u) = 0 \tag{3.10}$$

$$u(x,0) = u_0(x) \tag{3.11}$$

3.2.1 ENO/WENO Approximation - For Spatial Discretization

Here, brief explanation will be given. Note that we consider only for Hamilton-Jacobi ENO/WENO, here. For more detail, see [18, 19, 29].

ENO Approximation

In [33], they introduced the idea of essentially nonoscillatory (ENO) polynomial interpolation of data for the numerical solution of conservation laws.

Their basic idea was to compute numerical flux functions using the smoothest possible polynomial interpolants. In [6], they realized that Hamilton-Jacobi equations in one spatial dimension are integrals of conservation laws. They used this fact to extend the ENO method for the numerical discretization of conservation laws to Hamilton-Jacobi equations such as equation (3.10). This Hamilton-Jacobi ENO method allows one to extend first-order accurate upwind differencing to higher-order spatial accuracy by providing better numerical approximations to ϕ_x^- or ϕ_x^+.
Basic idea of Hamilton-Jacobi ENO method is to use the smoothest polynomial interpolation to find ϕ and then differentiate to get ϕ_x. Similar notation with Newton Polynomial Interpolation, the zeroth divided differences of ϕ are defined at the grid nodes and defined by

$$D_i^0\phi = \phi_i \tag{3.12}$$

at each grid node i. The first divided differences of ϕ are defined midway between grid nodes as

$$D_{i+\frac{1}{2}}^1\phi = \frac{D_{i+1}^0\phi - D_i^0}{\triangle x} \tag{3.13}$$

where we are assuming that $\triangle x$ is uniform. Note that $D_{i-\frac{1}{2}}^1\phi = (D^-\phi)_i$ and $D_{i+\frac{1}{2}}^1\phi = (D^+\phi)_i$, i.e., the first divided differences, are the backward and forward difference approximations to the derivatives. The second divided differences are defined at the grid nodes as

$$D_i^2\phi = \frac{D_{i+\frac{1}{2}}^1\phi - D_{i-\frac{1}{2}}^1}{2\triangle x} \tag{3.14}$$

while the third divided differences

$$D_{i+\frac{1}{2}}^3\phi = \frac{D_{i+1}^2\phi - D_i^2}{3\triangle x} \tag{3.15}$$

are defined midway between the grid nodes.
The divided differences are used to reconstruct a polynomial of the form

$$\phi(x) = Q_0(x) + Q_1(x) + Q_2(x) + Q_3(x) \tag{3.16}$$

that can be differentiated and evaluated at x_i to find $(\phi_x^-)_i$ and $(\phi_x^+)_i$. That is, we use

$$\phi_x(x_i) = Q_1'(x_i) + Q_2'(x_i) + Q_3'(x_i) \tag{3.17}$$

to define $(\phi_x^-)_i$ and $(\phi_x^+)_i$. Note that the constant $Q_0(x)$ term vanishes upon differentiation.

To find ϕ_x^- we start with $k = i-1$, and to find ϕ_x^+ we start with $k = i$. Then we define

$$Q_1(x) = (D^1_{k+\frac{1}{2}}\phi)(x - x_i) \tag{3.18}$$

so that

$$Q_1'(x) = D^1_{k+\frac{1}{2}}\phi \tag{3.19}$$

implying that the contribution from $Q_1'(x_i)$ in Eqn (3.17) is the backward difference in the case of ϕ_x^- and the forward difference in the case of ϕ_x^+. Improvements are obtained by including the $Q_2'(x_i)$ and $Q_3'(x_i)$ terms in Eqn (3.17), leading to second- and third-order accuracy, repectively.

Looking at the divided difference table and noting that $D^1_{k+\frac{1}{2}}\phi$ was chosen for first-order accuracy, we have two choices for the second-order accurate correction. We could include the next point to the left and use $D^2_k\phi$, or we could include the next point to the right and use $D^2_{k+1}\phi$. The key observation is that smooth slowly varying data tend to produce small numbers in divided difference tables, while discontinuous or quickly varying data tend to produce large numbers in divided difference tables. This is obvious in the sense that the differences measure variation in the data.

Comparing $|D^2_k\phi|$ to $|D^2_{k+1}\phi|$ indicates which of the polynomial interpolants has more variation. We would like to avoid interpolating near large variations such as discontinuities or steep gradients, since they cause overshoots in the interpolating function, leading to numerical errors in the approximation of the derivative. Thus, if $|D^2_k\phi| \leq |D^2_{k+1}\phi|$, we set $c = D^2_k\phi$ and $k^* = k-1$; otherwise, we set $c = D^2_{k+1}\phi$ and $k^* = k$. Then we define

$$Q_2(x) = c(x - x_k)(x - x_{k+1}) \tag{3.20}$$

so that

$$Q_2'(x_i) = c(2(i-k)-1)\triangle x \tag{3.21}$$

is the second-order accurate correction to the approximation of ϕ_x in Eqn (3.17). If we stop here, i.e., omitting the Q_3 term, we have a second-order accurate method for approximating ϕ_x^- and ϕ_x^+. Note that k^* has not yet been used. It is defined below for use in calculating the third-order accurate correction.

Similar to the second-order accurate correction, the third-order accurate correction is obtained by comparing $|D^3_{k+\frac{1}{2}}\phi|$ and $|D^3_{k+\frac{3}{2}}\phi|$. If $|D^3_{k+\frac{1}{2}}\phi| \leq |D^3_{k+\frac{3}{2}}\phi|$, we set $c^* = D^3_{k^*+\frac{1}{2}}\phi$; otherwise, we set $c^* = D^3_{k^*+\frac{3}{2}}\phi$

$$Q_3(x) = c^*(x - x_{k^*})(x - x_{k^*+1})(x - x_{k^*+2}) \tag{3.22}$$

so that

$$Q_3'(x) = c^*(3(i - k^*)^2 - 6(i - k^*) + 2)(\triangle x)^2 \tag{3.23}$$

is the third-order accurate correction to the approximation of ϕ_x in Eqn (3.17).

WENO Approximation

WENO (Weighted ENO) approximation is a convex combination of the ENO approximations. Assume that we have a 3rd order ENO approximation of $(\phi_x^-)_i$. Then candidates for $(\phi_x^-)_i$ are ϕ_x^1, ϕ_x^2, ϕ_x^3. Then WENO approximation of $(\phi_x^-)_i$ is

$$\phi_x = \omega_1\phi_x^1 + \omega_2\phi_x^2 + \omega_3\phi_x^3 \tag{3.24}$$

where $0 \leq \omega_k \leq 1$ are the weights with $\omega_1 + \omega_2 + \omega_3 = 1$. It is proved that weights of $\omega_1 = 0.1, \omega_2 = 0.6, \omega_3 = 0.3$ give the optimal fifth-order accurate approximation to ϕ_x in smooth regions. In nonsmooth regions this optimal weighting can be inaccurate, so instead we can use ENO type approximation for accuracy in nonsmooth regions.

Not choosing this region dependent weights, we can use the smooth indicator to define the weights. For $(\phi_x^-)_i$, define $v_1 = D^-\phi_{i-2}, v_2 = D^-\phi_{i-1}, v_3 = D^-\phi_i, v_4 = D^-\phi_{i+1}, v_5 = D^-\phi_{i+2}$ where $D^-\phi_i = \frac{\phi_i - \phi_{i-1}}{\triangle x}$. Then the smooth

indicators is defined

$$S_1 = \frac{13}{12}(v_1 - 2v_2 + v_3)^2 + \frac{1}{4}(v_1 - 4v_2 + 3v_3)^2, \tag{3.25}$$

$$S_2 = \frac{13}{12}(v_2 - 2v_3 + v_4)^2 + \frac{1}{4}(v_2 - v_4)^2, \tag{3.26}$$

$$S_3 = \frac{13}{12}(v_3 - 2v_4 + v_5)^2 + \frac{1}{4}(3v_3 - 4v_4 + v_5)^2, \tag{3.27}$$

Using these smooth indicators, define

$$\alpha_1 = \frac{0.1}{(S_1 + \epsilon)^2}, \tag{3.28}$$

$$\alpha_2 = \frac{0.6}{(S_2 + \epsilon)^2}, \tag{3.29}$$

$$\alpha_3 = \frac{0.9}{(S_3 + \epsilon)^2}, \tag{3.30}$$

where

$$\epsilon = 10^{-6} max\{v_1^2, v_2^2, v_3^2, v_4^2, v_5^2\} + 10^{-99} \tag{3.31}$$

Finally, we can obtain the weights by normalizing α_k

$$\omega_1 = \frac{\alpha_1}{\alpha_1 + \alpha_2 + \alpha_3}, \tag{3.32}$$

$$\omega_2 = \frac{\alpha_2}{\alpha_1 + \alpha_2 + \alpha_3}, \tag{3.33}$$

$$\omega_3 = \frac{\alpha_3}{\alpha_1 + \alpha_2 + \alpha_3} \tag{3.34}$$

Note that when the S_k are small enough to be dominated by ϵ, then the weights are $\omega_1 = 0.1, \omega_2 = 0.6, \omega_3 = 0.3$ which are same as smooth regional case. This is why the S_k are called smooth indicators.

3.2.2 TVD Runge Kutta Scheme - For Time Discretization

Most methods used in practice do not require that the user explicitly calculate higher order derivatives. Instead a higher order finite difference approximation is designed that typically models these terms automatically.

A multistep method can achieve high accuracy by using high order polynomial interpolation through several previous values of the solution and its derivative. To achieve the same effect with a 1-step method it is typically necessary to use a *multistage* method, where intermediate values of the solution and its derivative are generated and used within a single time step. Here, we are introducing Runge-Kutta method, which is famous multistage method, as applying to general advection equation. A two-stage explicit Runge-Kutta method is given by

$$u^* = u^n + \frac{1}{2}kf(u^n) \tag{3.35}$$

$$u^{n+1} = u^n + kf(u^*) \tag{3.36}$$

where $k = \triangle t$. In the first stage and intermediate value is generated that approximates $u(t_{n+1/2})$ via Euler's method. In the second step the function f is evaluated at this midpoint to estimate the slope over the full time step. Since this now looks like a centered approximation to the derivative we might hope for second order accuracy.
Combining the two steps above, we can rewrite the method as

$$u^{n+1} = u^n + kf(u^n + \frac{1}{2}kf(u^n)) \tag{3.37}$$

Viewed this way, this is clearly a 1-step explicit method. The truncation error is

$$\tau^n = \frac{1}{k}(u(t_{n+1}) - u(t_n)) - f(u(t_n) + \frac{1}{2}kf(u(t_n))) \tag{3.38}$$

Note that

$$\begin{aligned} f(u(t_n) + \frac{1}{2}kf(u(t_n))) &= f(u(t_n) + \frac{1}{2}ku'(t_n)) \\ &= f(u(t_n)) + \frac{1}{2}ku'(t_n)f'(u(t_n)) + \frac{1}{8}k^2(u'(t_n))^2f''(u(t_n)) + \cdots \end{aligned}$$

Since $f(u(t_n)) = u'(t_n)$ and differentiating gives $f'(u)u' = u''$, we obtain

$$f(u(t_n) + \frac{1}{2}kf(u(t_n))) = u'(t_n) + \frac{1}{2}ku''(t_n) + O(k^2)$$

Using this in (3.37) gives

$$\tau^n = \frac{1}{k}(ku'(t_n) + \frac{1}{2}k^2u''(t_n) + O(k^3)) - (u'(t_n) + \frac{1}{2}ku''(t_n) + O(k^2)) \quad = O(k^2)$$

and the method is second order accurate.
The method given by (3.35) (3.36) can be extended to nonautonomous equations of the form $u'(t) = f(u(t), t)$:

$$u^* = u^n + \frac{1}{2}kf(u^n, t_n) \tag{3.39}$$

$$u^{n+1} = u^n + kf(u^*, t_n + \frac{1}{2}) \tag{3.40}$$

This is again second order accurate, but it is slightly more complicated since Taylor series in two variables must be used.
One simple higher order Runge-Kutta method is the fourth order four-stage method given by

$$\begin{aligned} U_1 &= u^n, \\ U_2 &= u^n + \frac{1}{2}kf(U_1, t_n), \\ U_3 &= u^n + \frac{1}{2}kf(U_2, t_n + \frac{k}{2}), \\ U_4 &= u^n + kf(U_3, t_n + \frac{k}{2}), \end{aligned}$$

$$u^{n+1} = u^n + \frac{k}{6}[f(U_1, t_n) + 2f(U_2, t_n + \frac{k}{2}) + 2f(U_3, t_n + \frac{k}{2}) + f(U_4, t_n + k)]. \tag{3.41}$$

A general r-stage Runge-Kutta method has the form

$$U_1 = u^n + k\sum_{j=1}^{r} a_{1j}f(U_j, t_n + c_jk),$$

$$U_2 = u^n + k\sum_{j=1}^{r} a_{2j}f(U_j, t_n + c_jk),$$

$$\begin{aligned}
&\vdots \\
U_r &= u^n + k\sum_{j=1}^{r} a_{rj} f(U_j, t_n + c_j k), \\
u^{n+1} &= u^n + k\sum_{j=1}^{r} b_j f(U_j, t_n + c_j k)
\end{aligned} \tag{3.42}$$

Consistency requires

$$\begin{aligned}
\sum_{j=1}^{r} a_{ij} &= c_i, \quad i = 1, 2, \cdots, r, \\
\sum_{j=1}^{r} b_j &= 1,
\end{aligned}$$

If these conditions are satisfied, then the method will be at least first order accurate.
The coefficients for a Runge-Kutta method are often displayed in a so-called Butcher tableau:

$$\begin{array}{c|ccc}
c_1 & a_{11} & \cdots & a_{1r} \\
\vdots & \vdots & & \vdots \\
c_r & a_{r1} & \cdots & a_{rr} \\
\hline
 & b_1 & \cdots & b_r
\end{array}$$

For example, the fourth order Runge-Kutta method given in (3.41) has the following tableau(entries not shown are all 0):

$$\begin{array}{c|cccc}
0 & & & & \\
1/2 & 1/2 & & & \\
1/2 & 0 & 1/2 & & \\
1 & 0 & 0 & 1 & \\
\hline
 & 1/6 & 1/3 & 1/3 & 1/6
\end{array}$$

An important class of Runge-Kutta methods consists of the *explicit methods* for each $a_{ij} = 0$ for $j \geq i$. For an explicit method, the elements on and above the diagonal in the a_{ij} portion of the Butcher tableau are all equal to zero,

as, for example, with the fourth order method displayed above. With an explicit method, each of the U_i values is computed using only the previously computed U_j.

3.2.3 Semi-Lagrangian/BDF mixed Scheme

Unlike ENO/WENO scheme, in [31], They use Semi-Lagrangian/Backward Difference Formula(BDF) mixed method for releasing CFL time step. This method transforms (3.10) into

$$\frac{3u^{n+1} - 4u_d^n + u_d^{n-1}}{2\triangle t} = 0 \tag{3.43}$$

where $u_d^n = u^n(x^n)$ and $u_d^{n-1} = u^{n-1}(x^{n-1})$ indicate that the u^n and u^{n-1} velocities are computed at the departure locations by semi-lagrangian manner. Note that if we delete subscript d from u_d, then (3.43) is just a 2nd order BDF for the time derivatives, which is implicit method in time.
In [31], u_d^n and u_d^{n-1} are defined as follows to obtain second order in time and space.

$$u_d^n = u^n(x^{n+1} - \triangle t u^{n+\frac{1}{2}}(x^{n+1} - \frac{1}{2}\triangle t u^n(x^{n+1})), \tag{3.44}$$

where $u^{n+\frac{1}{2}} = \frac{3}{2}u^n - \frac{1}{2}u^{n-1}$, and

$$u_d^n = u^n(x^{n+1} - 2\triangle t u^n(x^{n+1} - \triangle t u^n(x^{n+1})), \tag{3.45}$$

However, it is known that computing u_d^n and u_d^{n-1} is expensive than necessary by Taylor series analysis of above process. Hence, in practice, the following simplified discretized version is used.

$$u_d^n = u^n(x^{n+1} - \triangle t u^n(x^{n+1})) \tag{3.46}$$

$$u_d^{n-1} = u^{n-1}(x^{n+1} - 2\triangle t u^{n-1}(x^{n+1})) \tag{3.47}$$

Remark Since BDF is a multistep method, the first step must be taken by reasonable way. We can choose one order lower way for this one time step. So using simple backward Euler method, we can have

$$\frac{u^{n+1} - u_d^n}{\triangle t} = 0 \tag{3.48}$$

as a first step discretization.

3.3 Projection Term

3.3.1 Poisson's Equation

Solving Poisson's equation is crucial for the incompressible Navier-Stokes equation equipped with a projection method. The Poisson's equation for this problem is reformulated as follows:

$$\nabla \cdot (\frac{1}{\rho}\nabla p^{n+1}) = \frac{1}{\triangle t}\nabla \cdot \mathbf{u}^* \tag{3.49}$$

with the Neumann boundary condition

$$(\frac{\partial p}{\partial \mathbf{n}})^{n+1}_{\partial\Omega} = -\frac{\rho}{\triangle t}(\mathbf{u}^{n+1} - \mathbf{u}^*) \cdot \mathbf{n} \tag{3.50}$$

Because the solution of this Neumann problem is independent with the choice of $\mathbf{u}^*$, we can select $\mathbf{u}^* = \mathbf{u}^{n+1}$ for simplicity; that is,

$$(\frac{\partial p}{\partial \mathbf{n}})^{n+1}_{\partial\Omega} = 0 \tag{3.51}$$

The condition of compatibility for the Neumann problem is

$$0 = \int_{\partial\Omega} (\frac{\partial p}{\partial \mathbf{n}})^{n+1} d(\partial\Omega) = -\frac{\rho}{\triangle t}\int_{\partial\Omega} (\mathbf{u}^{n+1} - \mathbf{u}^*)\cdot\mathbf{n} d(\partial\Omega) = -\frac{\rho}{\triangle t}\int_{\Omega} (\nabla\cdot\mathbf{u}^{n+1} - \nabla\cdot\mathbf{u}^*) d\Omega \tag{3.52}$$

that is,

$$\int_{\Omega} \nabla \cdot \mathbf{u}^* d\Omega = \int_{\Omega} \nabla \cdot \mathbf{u}^{n+1} d\Omega \ = 0 \tag{3.53}$$

It is important for the discretization with respect to space to conserve the preceding compatibility condition.
This is the homogeneous Neumann Poisson problem, which is a singular system, $Ax = b$, in discretized form. Generally, when we solve singular system with PCG, some special treatment must be needed. By noting that e, the

vector of all ones, is a basis of null space of A, we solved $(A + \frac{1}{N}ee^t)y = b$, where N is the number of row, and orthogonalize y with respect to e to have real solution x. This modification guarantees null space of A is shifted away and we can find the solution (unique upto constant) on the restricted space of the Krylov sequences of CG Span.

3.3.2 Variable Coefficient Poisson's Equation with Jump Condition: Surface Tension Effect Considered

Instead of using a continuous surface tension (CSF) model [16], we use a boundary condition capturing (BCC) model for solving pressure [23], [21]. In the case of a smoothed out viscosity, the jump condition in pressure at interface Γ is known to be $[p]_\Gamma = \sigma\kappa$, where

$$[p]_\Gamma = p_{water} - p_{oil}$$

σ is the coefficient of surface tension and κ is the curvature of the interface. In dimensionless form, this can be modified, $[p] = \frac{1}{We}\kappa$, where $We = \frac{\rho_1 R V_0^2}{\sigma}$ is the Weber number. Then, Eq. (3.49) can be written as follows:

$$\nabla \cdot (\beta \nabla p^{n+1}) = \frac{1}{\triangle t}\nabla \cdot \mathbf{u}^* \tag{3.54}$$

$$(\frac{\partial p}{\partial \mathbf{n}})^{n+1}_{\partial\Omega} = 0,\ [p]_\Gamma = \sigma\kappa$$

where $\beta = \frac{1}{\rho}$. Because the BCC method can be extended in dimension-by-dimension fashion, we present only one case to solve this. Assume that interface Γ is located between $x_{i,j,k}$ and $x_{i-1,j,k}$, $\phi_{i,j,k} < 0$, and $\phi_{i-1,j,k} > 0$. Then, the discretized version of Eq. (3.54) is as follows:

$$[\beta_{i+1/2,j,k}(\frac{p^-_{i+1,j,k} - p^-_{i,j,k}}{\triangle x}) - \beta_{i-1/2,j,k}(\frac{p^-_{i,j,k} - p^+_{i-1,j,k}}{\triangle x})]/\triangle x +$$

$$[\beta_{i,j+1/2,k}(\frac{p_{i,j+1,k} - p_{i,j,k}}{\triangle y}) - \beta_{i,j-1/2,k}(\frac{p_{i,j,k} - p_{i,j-1,k}}{\triangle y})]/\triangle y +$$

$$[\beta_{i,j,k+1/2}(\frac{p_{i,j,k+1}-p_{i,j,k}}{\triangle z})-\beta_{i,j,k-1/2}(\frac{p_{i,j,k}-p_{i,j,k-1}}{\triangle z})]/\triangle z=\frac{1}{\triangle t}(\nabla\cdot\mathbf{u}^*)_{i,j,k} \tag{3.55}$$

In the preceding formula, $\frac{\partial}{\partial x}(\beta\frac{\partial p}{\partial x})$ has discretization with mixed sign on pressure, p. This mixing sign degrades the numerical solution of the PDE. Hence, the sign is unified by using the jump condition on p, $p^- = p^+ - (\sigma\kappa)_\Gamma$.

$$[\beta_{i+1/2,j,k}(\frac{p^-_{i+1,j,k}-p^-_{i,j,k}}{\triangle x})-\beta_{i-1/2,j,k}(\frac{p^-_{i,j,k}-p^+_{i-1,j,k}}{\triangle x})]/\triangle x$$

$$\longrightarrow [\beta_{i+1/2,j,k}(\frac{p^-_{i+1,j,k}-p^-_{i,j,k}}{\triangle x})-\beta_{i-1/2,j,k}(\frac{p^-_{i,j,k}-p^-_{i-1,j,k}}{\triangle x})]/\triangle x$$

$$\longrightarrow [\beta_{i+1/2,j,k}(\frac{p^-_{i+1,j,k}-p^-_{i,j,k}}{\triangle x})-\beta_{i-1/2,j,k}(\frac{p^-_{i,j,k}-(p^+_{i,j,k}-(\sigma\kappa)_\Gamma)}{\triangle x})]/\triangle x$$

Here, $(\sigma\kappa)_\Gamma$ can be obtained by using interpolation with subcell resolution.

$$(\sigma\kappa)_\Gamma=\frac{(\sigma\kappa)_{i,j,k}|\phi_{i-1,j,k}|+(\sigma\kappa)_{i-1,j,k}|\phi_{i,j,k}|}{|\phi_{i-1,j,k}|+|\phi_{i,j,k}|} \tag{3.56}$$

Substituting this into Eq. (3.55) and reorganizing the equation, we have

$$[\beta_{i+1/2,j,k}(\frac{p_{i+1,j,k}-p_{i,j,k}}{\triangle x})-\beta_{i-1/2,j,k}(\frac{p_{i,j,k}-p_{i-1,j,k}}{\triangle x})]/\triangle x+[\beta_{i,j+1/2,k}(\frac{p_{i,j+1,k}-p_{i,j,k}}{\triangle y})$$

$$-\beta_{i,j-1/2,k}(\frac{p_{i,j,k}-p_{i,j-1,k}}{\triangle y})]/\triangle y \;+\; [\beta_{i,j,k+1/2}(\frac{p_{i,j,k+1}-p_{i,j,k}}{\triangle z})$$

$$-\beta_{i,j,k-1/2}(\frac{p_{i,j,k}-p_{i,j,k-1}}{\triangle z})]/\triangle z=\frac{1}{\triangle t}(\nabla\cdot\mathbf{u}^*)_{i,j,k}+\frac{\beta_{i-1/2,j,k}(\sigma\kappa)_\Gamma}{(\triangle x)^2} \tag{3.57}$$

It is important for this modification to never degrade the approximated solution or change the matrix; that is, it should preserve the SPD system. Thus, we can apply the PCG method for solving this formula.
However, This approach has a limitation that it is assumed that $(\frac{\partial p}{\partial \mathbf{n}})_\Gamma = 0$, which is physically not exact. That is, depending on problems, $(\frac{\partial p}{\partial \mathbf{n}})_\Gamma$ cannot be negligible. More exact method can be found in [32], Immersed Interface Method (IIM), which gives a nonsymmetric discretization.

3.3.3 Preconditioned Conjugate Gradient Method

As mentioned above, we must solve the linear system (3.57) for obtaining a pressure. Among several methods, this thesis introduce the Preconditioned Conjugate Gradient Method, briefly. We refer [30] to reader for more detail.

Conjugate Gradient(CG) Method

The Conjugate Gradient Method is a method which tries to find the minimum value of the following *quadratic form*.

$$f(x) = \frac{1}{2}x^T Ax - b^T x + c \tag{3.58}$$

where A is a matrix, x and b are vectors, and c is a scalar constant. Before proceeding further, let us give a definition of a positive-definite matrix.

Definition 3.3.1. *A symmetric $n \times n$ matrix A is said to be* ***positive-definite*** *if $x^T Ax$ is positive for every non-zero column vector x of n real number.*

Now, assume that A is symmetric and positive-definite in (3.58). Then, quadratic form (3.58) will be shaped like a paraboloid bowl. This implies that given quadratic form has a minimum value which satisfies $f'(x) = 0$ where the *gradient* of a quadratic form is defined as

$$f'(x) = \begin{pmatrix} \frac{\partial}{\partial x_1} f(x) \\ \frac{\partial}{\partial x_2} f(x) \\ \vdots \\ \frac{\partial}{\partial x_n} f(x) \end{pmatrix} \tag{3.59}$$

Applying (3.59) to $f'(x)$, we can have

$$f'(x) = \frac{1}{2}A^T x + \frac{1}{2}Ax - b = 0 \tag{3.60}$$

Since A is symmetric, (3.60) reduces to

$$f'(x) = Ax - b = 0 \tag{3.61}$$

which equivalent to the system $Ax = b$.
So, we can have a solution of (3.57) to have a minimum value in corresponding quadratic form. Basic approach in finding a minimum value is to define a set of orthogonal *search direction* $d_{(0)}, d_{(1)}, \ldots, d_{(n-1)}$. From starting point $x_{(i)}$, find the next point $x_{(i+1)}$ such as

$$x_{(i+1)} = x_{(i)} + \alpha_{(i)} d_{(i)} \tag{3.62}$$

To find the value of $\alpha_{(i)}$, make use of the fact that the error $e_{(i+1)} = x_{(i)} - x$ must be orthogonal to $d_{(i)}$. Then we have

$$d_{(i)}^T e_{(i+1)} = 0 \tag{3.63}$$

$$d_{(i)}^T (e_{(i)} + \alpha_{(i)} d_{(i)}) = 0 \qquad from \quad (3.62) \tag{3.64}$$

$$\alpha_{(i)} = -\frac{d_{(i)}^T e_{(i)}}{d_{(i)}^T d_{(i)}} \tag{3.65}$$

But, this is impossible since there is no way of knowing the error $e_{(i)}$. For this reason, we introduce slightly modified orthogonal relation, A-orthogonality. Two vectors $d_{(i)}$ and $d_{(j)}$ are *A-orthogonal*, or *conjugate*, if

$$d_{(i)}^T A d_{(j)} = 0 \tag{3.66}$$

From this new orthogonality relation, our new requirement is that $e_{(i+1)}$ be A-orthogonal to $d_{(i)}$. That is,

$$\frac{d}{d\alpha} f(x_{(i+1)}) = 0 \tag{3.67}$$

$$f'(x_{(i+1)})^T \frac{d}{d\alpha} x_{(i+1)} = 0 \tag{3.68}$$

$$-r_{(i+1)}^T d_{(i)} = 0 \tag{3.69}$$

$$d_{(i)}^T A e_{(i+1)} = 0 \tag{3.70}$$

Following the derivation of (3.65), we have

$$\alpha_{(i)} = -\frac{d_{(i)}^T A e_{(i)}}{d_{(i)}^T A d_{(i)}} \tag{3.71}$$

$$= \frac{d_{(i)}^T r_{(i)}}{d_{(i)}^T A d_{(i)}} \tag{3.72}$$

where $r_{(i)} = b - Ax_{(i)} = -Ae_{(i)}$ is a residual vector.
Now, all we need for finding minimum is a set of A-orthogonal search direction $\{d_{(i)}\}$. There is a well-known process called a *conjugate Gram-Schmidt process*, which is extension version of Gram-Schmidt orthogonalization process.
Suppose that a set of n linearly independent vectors $u_0, u_1, \cdots, u_{n-1}$ are given. We want to find $d_{(i)}$ such as

$$d_{(i)} = u_i + \sum_{k=0}^{i-1} \beta_{ik} d_{(k)} \tag{3.73}$$

where β_{ik} are defined for $i > k$ and $d_{(0)} = u_0$. Now, we have

$$d_{(i)}^T A d_{(j)} = u_i^T A d_{(j)} + \sum_{k=0}^{i-1} \beta_{ik} d_{(k)}^T A d_{(j)} \tag{3.74}$$

$$0 = u_i^T A d_{(j)} + \beta_{ij} d_{(j)}^T A d_{(j)}, \qquad i > j \tag{3.75}$$

$$\beta_{ij} = -\frac{u_i^T A d_{(j)}}{d_{(j)}^T A d_{(j)}} \tag{3.76}$$

Note that (3.75) used A-orthogonality of d vectors.
Finally, we can define the Conjugate Gradient Method. *Conjugate Gradient(CG) Method* is exactly same method with above discussion except an A-orthogonal set definition. In CG Method, we define an A-orthogonal set as a *Krylov suvspace*

$$\begin{aligned} D_i &= span\{d_{(0)}, Ad_{(0)}, A^2 d_{(0)}, \cdots, A^{(i-1)} d_{(0)}\} \\ &= span\{r_{(0)}, Ar_{(0)}, A^2 r_{(0)}, \cdots, A^{(i-1)} r_{(0)}\} \end{aligned}$$

Now, we can express explicit algorithm of CG method. First, finding β's as

follows:

$$r_{(i)}^T r_{(j+1)} = r_{(i)}^T r_{(j)} - \alpha_{(j)} r_{(i)}^T A d_{(j)} \tag{3.77}$$

$$\alpha_{(j)} r_{(i)}^T A d_{(j)} = r_{(i)}^T r_{(j)} - r_{(i)}^T r_{(j+1)} \tag{3.78}$$

$$r_{(i)} A d_{(j)} = \begin{cases} \frac{1}{\alpha_{(i)}} r_{(i)}^T r_{(i)}, & i = j, \\ -\frac{1}{\alpha_{(i)}} r_{(i)}^T r_{(i)}, & i = j+1, \\ 0, & \text{otherwise.} \end{cases} \tag{3.79}$$

$$\therefore \beta_{ij} = \begin{cases} \frac{1}{\alpha_{(i-1)}} \frac{r_{(i)}^T r_{(i)}}{d_{(i-1)}^T A d_{(i-1)}}, & i = j+1, \\ 0, & i > j+1. \end{cases} \tag{3.80}$$

From (3.80), we can use the abbreviation $\beta_{(i)} = \beta_{i,i-1}$. Simplifying gives

$$\beta(i) = \frac{r_{(i)}^T r_{(i)}}{d_{(i-1)}^T r_{(i-1)}} \tag{3.81}$$

$$= \frac{r_{(i)}^T r_{(i)}}{r_{(i-1)}^T r_{(i-1)}} \tag{3.82}$$

Finally, the method of Conjugate Gradient is

$$d_{(0)} = r_{(0)} = b - A x_{(0)}, \tag{3.83}$$

$$\alpha_{(i)} = \frac{r_{(i)}^T r_{(i)}}{d_{(i)}^T A d_{(i)}}, \tag{3.84}$$

$$x_{(i+1)} = x_{(i)} + \alpha_{(i)} d_{(i)}, \tag{3.85}$$

$$r_{(i+1)} = r_{(i)} - \alpha_{(i)} A d_{(i)}, \tag{3.86}$$

$$\beta_{(i+1)} = \frac{r_{(i+1)}^T r_{(i+1)}}{r_{(i)}^T r_{(i)}}, \tag{3.87}$$

$$d_{(i+1)} = r_{(i+1)} + \beta_{(i+1)} d_{(i)}. \tag{3.88}$$

Preconditioned Conjugate Gradient(PCG) Method

From the convergence analysis on CG method [30], it is important to improve condition number of matrix for fast convergence of method. Preconditioning

is a technique for improving the condition number of a matrix. Suppose that M is a symmetric, positive-definite matrix that approximates A, but is easy enough to have its own inverse. Modifying the given linear system $Ax = b$ as

$$M^{-1}Ax = M^{-1}b \tag{3.89}$$

gives fast convergence if $\kappa(M^{-1}A) \ll \kappa(A)$, that is, if the eigenvalues of $M^{-1}A$ are better clustered than those of A. However, the underlying problem in this modification is that $M^{-1}A$ is not generally symmetric nor definite, even if M and A are.
So, we try to resolve this using the following fact: Every symmetric, positive-definite matrix M has a decomposition (don't need to be unique) $EE^T = M$. Using the simple fact that $M^{-1}A$ and $E^{-1}AE^{-T}$ have the same eigenvalues, we can transform the given linear system $Ax = b$ as

$$E^{-1}AE^{-T}\hat{x} = E^{-1}b \tag{3.90}$$

$$\hat{x} = E^T x \tag{3.91}$$

In (3.90), we solve first for $\hat{x}$, the for x. Since $E^{-1}AE^{-T}$ is symmetric and positive-definite, $\hat{x}$ can be found by CG method. The CG method to solve (3.90) is called *Transformed Preconditioned Conjugate Gradient Method*:

$$\hat{d}_{(0)} = \hat{r}_{(0)} = E^{-1}b - E^{-1}AE^{-T}\hat{x}_{(0)} \tag{3.92}$$

$$\alpha_{(i)} = \frac{\hat{r}_{(i)}^T \hat{r}_{(i)}}{\hat{d}_{(i)}^T E^{-1}AE^{-T}\hat{d}_{(i)}} \tag{3.93}$$

$$\hat{x}_{(i+1)} = \hat{x}_{(i)} + \alpha_{(i)}\hat{d}_{(i)} \tag{3.94}$$

$$\hat{r}_{(i+1)} = \hat{r}_{(i)} - \alpha_{(i)}E^{-1}AE^{-T}\hat{d}_{(i)} \tag{3.95}$$

$$\beta_{(i+1)} = \frac{\hat{r}_{(i+1)}^T \hat{r}_{(i+1)}}{\hat{r}_{(i)}^T \hat{r}_{(i)}} \tag{3.96}$$

$$\hat{d}_{(i+1)} = \hat{r}_{(i+1)} + \beta_{(i+1)}\hat{d}_{(i)} \tag{3.97}$$

The method has some disadvantage that E must be computed. However, simple modification can eliminate E. Setting $\hat{r}_{(i)} = E^{-1}r_{(i)}$ and $\hat{d}_{(i)} = E^T d_{(i)}$, and using the identities $\hat{x}_{(i)} = E^T x_{(i)}$ and $E^{-T}E^{-1} = M^{-1}$, we derive the

Untransformed Preconditioned Conjugate Gradient Method:

$$r_{(0)} = b - Ax_{(0)}, \tag{3.98}$$

$$d_{(0)} = M^{-1}r_{(0)}, \tag{3.99}$$

$$\alpha_{(i)} = \frac{r_{(i)}^T M^{-1} r_{(i)}}{d_{(i)}^T A d_{(i)}}, \tag{3.100}$$

$$x_{(i+1)} = x_{(i)} + \alpha_{(i)} d_{(i)}, \tag{3.101}$$

$$r_{(i+1)} = r_{(i)} - \alpha_{(i)} A d_{(i)}, \tag{3.102}$$

$$\beta_{(i+1)} = \frac{r_{(i+1)}^T M^{-1} r_{(i+1)}}{r_{(i)}^T M^{-1} r_{(i)}}, \tag{3.103}$$

$$d_{(i+1)} = M^{-1} r_{(i+1)} - \beta_{(i+1)} d_{(i)} \tag{3.104}$$

We called Untransformed Preconditioned Conjugate Gradient Method as Preconditioned Conjugate Gradient(PCG) Method.

3.4 Viscosity Term

The viscous term $\nabla \cdot (2\mu S)$ in Eq.(3.1) can be discretized by using central differencing. For example, in x directional velocity u,

$$(\frac{1}{Re}\frac{(2\mu u_x)_x + (\mu(u_y + v_x))_y + (\mu(u_z + w_x))_z}{\rho})_{i+1/2,j,k} \tag{3.105}$$

we have

$$((\mu(u_y + v_x))_y)_{i+1/2,j,k} = \frac{(\mu(u_y + v_x))_{i+1/2,j+1/2,k} - (\mu(u_y + v_x))_{i+1/2,j-1/2,k}}{\triangle y}$$

where

$$(u_y + v_x)_{i+1/2,j+1/2,k} = \frac{u_{i+1/2,j+1,k} - u_{i+1/2,j,k}}{\triangle y} + \frac{v_{i+1,j+1/2,k} - u_{i,j+1/2,k}}{\triangle x}$$

$$\mu_{i+1/2,j+1/2,k} = \frac{1}{4}(\mu_{i+1,j,k} + \mu_{i,j,k} + \mu_{i+1,j+1,k} + \mu_{i,j,k})$$

The other terms in Eq. (3.105) and the y, z directional velocities v, w can be discretized similarly.

3.4.1 Semi-Implicit Viscosity Solver

Depending on situations, There will be relatively large difference in viscosity between each phase. In these cases, viscous terms must be updated in a strictly small time step to ensure stability. This causes the entire simulation to be inefficient in terms of CFL time restrictions. To modify this, we introduce the unconditionally stable semi-implicit technique [25] for updating viscosity terms.
In the semi-implicit method, each velocity is updated only as an implicit or explicit term. That is,

$$\frac{u^* - u^n}{\triangle t} + (\mathbf{u}^n \cdot \nabla)u^n = \frac{1}{Re}\frac{(2\mu u_x^*)_x + (\mu(u_y^* + v_x^n))_y + (\mu(u_z^* + w_x^n))_z}{\rho^n} + F_1 \tag{3.106}$$

$$\frac{v^* - v^n}{\triangle t} + (\mathbf{u}^n \cdot \nabla)v^n = \frac{1}{Re}\frac{(\mu(u_y^n + v_x^*))_x + (2\mu v_y^*)_y + (\mu(v_z^* + w_y^n))_z}{\rho} + F_2 \tag{3.107}$$

$$\frac{w^* - w^n}{\triangle t} + (\mathbf{u}^n \cdot \nabla)w^n = \frac{1}{Re}\frac{(\mu(u_z^n + w_x^*))_x + (\mu(v_z^n + w_y^*))_y + (2\mu w_z^*)_z}{\rho} + F_3 \tag{3.108}$$

This modification provides the following systems:

- **u**-velocity

$$[I - \frac{\triangle t}{Re \cdot \rho}(\frac{\partial}{\partial x}(2\mu\frac{\partial}{\partial x}) + \frac{\partial}{\partial y}(\mu\frac{\partial}{\partial y}) + \frac{\partial}{\partial z}(\mu\frac{\partial}{\partial z}))]u^* = explicit\ term\ x \tag{3.109}$$

- **v**-velocity

$$[I - \frac{\triangle t}{Re \cdot \rho}(\frac{\partial}{\partial x}(\mu\frac{\partial}{\partial x}) + \frac{\partial}{\partial y}(2\mu\frac{\partial}{\partial y}) + \frac{\partial}{\partial z}(\mu\frac{\partial}{\partial z}))]v^* = explicit\ term\ y \tag{3.110}$$

- **w**-velocity

$$[I - \frac{\triangle t}{Re \cdot \rho}(\frac{\partial}{\partial x}(\mu\frac{\partial}{\partial x}) + \frac{\partial}{\partial y}(\mu\frac{\partial}{\partial y}) + \frac{\partial}{\partial z}(2\mu\frac{\partial}{\partial z}))]w^* = explicit\ term\ z \tag{3.111}$$

where

$$explicit\ term\ x = u^n - \triangle t(\mathbf{u}^n \cdot \nabla)u^n + \frac{\triangle t}{Re \cdot \rho}((\mu v_x^n)_y + (\mu w_x^n)_z) + \triangle t F_1$$

$$explicit\ term\ y = v^n - \triangle t(\mathbf{u}^n \cdot \nabla)v^n + \frac{\triangle t}{Re \cdot \rho}((\mu u_y^n)_x + (\mu w_y^n)_z) + \triangle t F_2$$

$$explicit\ term\ z = w^n - \triangle t(\mathbf{u}^n \cdot \nabla)w^n + \frac{\triangle t}{Re \cdot \rho}((\mu u_z^n)_x + (\mu v_z^n)_y) + \triangle t F_3$$

These systems are symmetric positive definite (SPD). Hence, we can solve these by using a fast algorithm such as the preconditioned conjugate gradient (PCG) method. A previous study [25] recommended a factorization scheme to solve these systems. When we simulate this element, we found that the semi-implicit solver requires less than 30 iterations, except for the first step because of the time step. This implies that the efficiency of the semi-implicit step is not degraded when we solve this part directly.

3.5 Boundary Conditions

In many cases, computational domain is not generally regular. So it is necessary to establish the boundary conditions for irregular domain. We introduce several technique to approximate boundary conditions for each term. For this reason, we introduce a level set function for boundary as ϕ_c.

3.5.1 Advection/Viscous Terms

Define the interpolated boundary level set $(\phi_c)_{i+1/2,j,k}$, $(\phi_c)_{i,j+1/2,k}$, $(\phi_c)_{i,j,k+1/2}$ as follows:

$$(\phi_c)_{i+1/2,j,k} = \frac{1}{2}((\phi_c)_{i,j,k} + (\phi_c)_{i-1,j,k}) \tag{3.112}$$

$$(\phi_c)_{i,j+1/2,k} = \frac{1}{2}((\phi_c)_{i,j,k} + (\phi_c)_{i,j-1,k}) \tag{3.113}$$

$$(\phi_c)_{i,j,k+1/2} = \frac{1}{2}((\phi_c)_{i,j,k} + (\phi_c)_{i,j,k-1}) \tag{3.114}$$

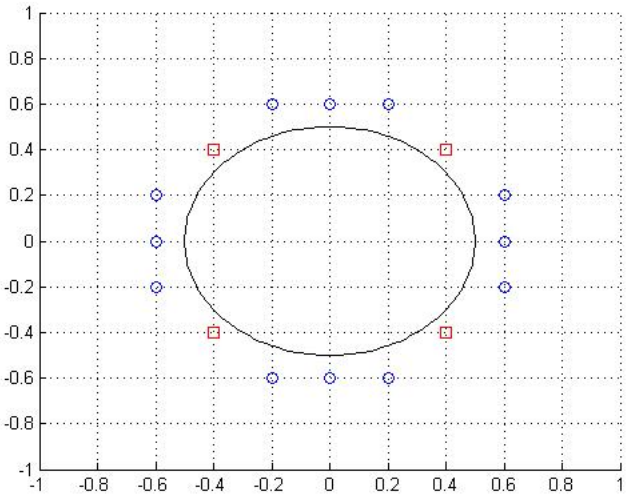


Figure 3.1: Points near the boundary of the domain. Blue circles are related only to one point inside the domain, whereas red squares are related to two points.

According to the sign of ϕ_c, we can determine whether a given domain is inside or outside. For example, consider the case when $(\phi_c)_{i+1/2,j,k} < 0$, $(\phi_c)_{i+3/2,j,k} > 0$, which is represented by blue circles in Fig. 1.
For velocity advection and viscous terms, define $u_{i+3/2,j,k}$ as

$$u_{i+3/2,j,k} = \frac{2u_\Gamma + (2\theta - 2)u_{i+1/2,j,k} + (-\theta^2 + 1)u_{i-1/2,j,k}}{\theta^2 + \theta} \tag{3.115}$$

where

$$\theta = \frac{|(\phi_c)_{i+1/2,j,k}|}{|(\phi_c)_{i+1/2,j,k}| + |(\phi_c)_{i+3/2,j,k}|} \tag{3.116}$$

This setup provides the velocity boundary condition on the pipe at second-order accuracy, which is consistent with choosing second-order accurate time integration. In the ENO 3rd method, we require more than one stencil point. Establish points on the stencil outside the domain as large values with large variation. For example, in the preceding case, establish $u_{i+5/2,j,k} = (10t)^{10}$, where $t = ((i + 5/2) - (i + 1/2))$.

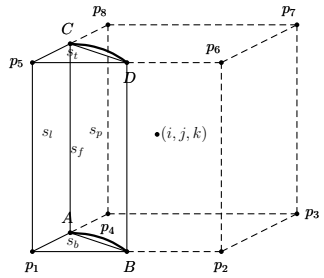


Figure 3.2: Cell near the boundary region.

In red squares, two different values can be assigned for one point. Instead of using these two values independently, we use bilinear interpolation to save memory in the CPU.

3.5.2 Projection Term

In projection step, we need to solve the Poisson equation with Neumann boundary condition:

$$(\frac{\partial p}{\partial n})_{\partial\Omega} = 0 \tag{3.117}$$

Here, we extended the direct discretization in [26] to three dimensional space. Assume that $\partial\Omega$ lies in the cell $C_{i,j,k}$. Now, applying divergence theorem to the integral form of (3.49) on $\Omega \cap C_{i,j,k}$ results in

$$\int_{\Omega\cap C_{i,j,k}} \nabla\cdot(\beta\nabla p^{n+1})dV = \int_{\partial(\Omega\cap C_{i,j,k})} \beta\nabla p^{n+1}\cdot\mathbf{n}dS = \int_{\Omega\cap C_{i,j,k}} \frac{1}{\triangle t}\nabla\cdot\mathbf{u}^* dV \tag{3.118}$$

where $\mathbf{n}$ is outward normal to interface.
As one example, consider a cell cut by the boundary pipe as shown in Fig.2. In this figure, polyhedron with bases p_5CD, p_1AB is $\Omega\cap C_{i,j,k}$. Approximating bases as $\triangle p_5CD$, $\triangle p_1AB$ and pipe region as rectangle $ABCD$ and applying boundary condition of no flow normal to pipe surface s_p, $\nabla p\cdot\mathbf{n} = 0$, we can

have the following discretization of lefthand side of Eqn (3.117).

$$\int_{\partial(\Omega\cap C_{i,j,k})} \beta\nabla p^{n+1}\cdot\mathbf{n}dS \approx \beta_{i-\frac{1}{2},j,k}p_l s_l+\beta_{i,j+\frac{1}{2},k}p_t s_t+\beta_{i,j-\frac{1}{2},k}p_b s_b+\beta_{i,j,k+\frac{1}{2}}p_f s_f \tag{3.119}$$

where

$$p_l = \frac{p_{i-1,j,k}-p_{i,j,k}}{\triangle x}, \quad p_t = \frac{p_{i,j+1,k}-p_{i,j,k}}{\triangle y}$$
$$p_b = \frac{p_{i,j-1,k}-p_{i,j,k}}{\triangle y}, \quad p_f = \frac{p_{i,j,k+1}-p_{i,j,k}}{\triangle z}$$

and s_l, s_t, s_b, s_f, s_p are the area of each surface designated in Fig 2. For the discretization of righthand side of (3.118), we can assume that numerical quantities of $\frac{1}{\triangle t}\nabla\cdot V$ is cell-wise constant. Since general finite difference discretization is used for cells in $\Omega-\partial\Omega$, this cell-wise constant assumption guarantee the consistency of discretization in whole domain. Hence the righthand side of (3.118) will be

$$\int_{\Omega\cap C_{i,j,k}} \frac{1}{\triangle t}\nabla\cdot\mathbf{u}^* dV \approx \frac{1}{\triangle t}(\nabla\cdot\mathbf{u}^*)_{i,j,k}\cdot Volume(\Omega\cap C_{i,j,k})$$

This boundary treatment technique preserves symmetry positivity of the corresponding matrix so we can apply efficient solver to solve poisson equation.

3.6 Reinitialization

Although Eq. (3.2) will move the level set $\phi = 0$ at the correct velocity, ϕ will no longer be a distance function (i.e., $|\nabla\phi| \neq 1$); ϕ can become irregular after a certain period of time. Maintaining ϕ as a distance function is essential for providing the interface with a width fixed in time. Computation of surface tension is difficult near a steep gradient in the distance function. The values for $\rho(\phi)$, especially for large density ratios, will be greatly distorted if $|\nabla\phi|$ is far from 1.

Conventional routines for reinitializing a distance function have to explicitly find the contour, ϕ=0, and reset ϕ at all points close to the front. An iteration method for reinitializing ϕ is described in the following.

Given a region, Ω^+, with $\phi \geq 0$ on Ω^+ and ϕ=0 on $\partial\Omega$, evolve the equation $\phi_t = 1 - |\nabla\phi|$ until ϕ reaches steady state. If ϕ is already close to a distance function, then one should not have to evolve too far in time.
Unfortunately, one still has to prescribe boundary conditions on $\partial\Omega^+$, which requires explicitly finding the interface. However, we can eliminate the problem of finding the interface. Consider the following function, $\phi_0(\mathbf{x})$, whose zero level set is the interface of two-phase: $\phi_0(\mathbf{x})$ need not to be a distance function. We will construct a function, $\phi(\mathbf{x})$, with the properties that its zero level set is the same as $\phi_0(\mathbf{x})$ and that ϕ is the signed normal distance to the interface. This is achieved by solving the following problem to steady state:

$$\phi_t = S(\phi_0)(1 - \sqrt{\phi_x^2 + \phi_y^2 + \phi_z^2}) \tag{3.120}$$

$$\phi(\mathbf{x}, 0) = \phi_0(\mathbf{x}) \tag{3.121}$$

where S is the signed function [8]. For numerical purposes, it is useful to smooth the sign function as

$$S_\epsilon(\phi_0) = \frac{\phi_0}{\sqrt{\phi_0^2 + \epsilon^2}} \tag{3.122}$$

However, Russo, G. and Smereka, P. [20] remarked that this approach can cause the zero levelset to move into the closest grid node because of inconsistency with upwinding approximation near interface. In order to fix this problem, they suggested the following modified version of Eqs.(3.120).

$$\phi_{i,j,k}^{n+1} = \begin{cases} \phi_{i,j,k}^n - \frac{\triangle t}{\triangle x}(sgn(\phi_{i,j,k}^0)|\phi_{i,j,k}^n| - D_{i,j,k}), & \text{if } (i,j,k) \in \Sigma_{\triangle x} \\ \phi_{i,j,k}^n - \triangle t \ sgn(\phi_{i,j,k}^0) G(\phi)_{i,j,k}, & \text{otherwise} \end{cases} \tag{3.123}$$

where

$$G(\phi)_{i,j,k} = \begin{cases} \sqrt{max(a_+^2, b_-^2) + max(c_+^2, d_-^2) + max(e_+^2, f_-^2)} - 1, & \text{if } \phi_{i,j,k}^0 > 0 \\ \sqrt{max(a_-^2, b_+^2) + max(c_-^2, d_+^2) + max(e_-^2, f_+^2)} - 1, & \text{if } \phi_{i,j,k}^0 < 0 \end{cases} \tag{3.124}$$

$$D_{i,j,k} = \frac{2\triangle\phi_{i,j,k}^0}{[(\phi_{i+1,j,k}^0 - \phi_{i-1,j,k}^0)^2 + (\phi_{i,j+1,k}^0 - \phi_{i,j-1,k}^0)^2 + (\phi_{i,j,k+1}^0 - \phi_{i,j,k-1}^0)^2]^{\frac{1}{2}}} \tag{3.125}$$

with

$$a \equiv D_x^- \phi_{i,j,k} = \frac{\phi_{i,j,k} - \phi_{i-1,j,k}}{\triangle x}, \quad b \equiv D_x^+ \phi_{i,j,k} = \frac{\phi_{i+1,j,k} - \phi_{i,j,k}}{\triangle x}$$

$$c \equiv D_y^- \phi_{i,j,k} = \frac{\phi_{i,j,k} - \phi_{i,j-1,k}}{\triangle x}, \quad d \equiv D_y^+ \phi_{i,j,k} = \frac{\phi_{i,j+1,k} - \phi_{i,j,k}}{\triangle x}$$

$$e \equiv D_z^- \phi_{i,j,k} = \frac{\phi_{i,j,k} - \phi_{i,j,k-1}}{\triangle x}, \quad f \equiv D_z^+ \phi_{i,j,k} = \frac{\phi_{i,j,k+1} - \phi_{i,j,k}}{\triangle x}$$

$$\Sigma_{\triangle x} = \{(i,j,k) | \phi^0_{i,j,k}\phi^0_{i-1,j,k} < 0 \quad or \quad \phi^0_{i,j,k}\phi^0_{i+1,j,k} < 0 \quad or \quad \phi^0_{i,j,k}\phi^0_{i,j+1,k} < 0$$
$$or \ \phi^0_{i,j,k}\phi^0_{i,j-1,k} < 0 \ \ or \ \phi^0_{i,j,k}\phi^0_{i,j,k+1} < 0 \ \ or \ \phi^0_{i,j,k}\phi^0_{i,j,k-1} < 0\}$$

Instead of using Eq. (3.125) we used a more robust formula based on Eq. (2.2.4) in [20] to avoid the denominator in Eq. (3.125) becoming very small. Since this approximation gives only first-order accurate in space, we choose the technique in [12] to apply 3rd ENO approximation on spatial derivatives.

3.7 CFL Condition

Except for the implicit term, which is unconditionally stable, a CFL time restriction may be needed for stability. In this thesis, the time condition is presented in dimensionless terms [21].

1. Advection Term

   - Dimensional Form

   $$A_c = \frac{|u|_{max}}{\triangle x} + \frac{|v|_{max}}{\triangle y} + \frac{|w|_{max}}{\triangle z}$$

   - Dimensionless Form

   $$A_c^* = \frac{|u^*|_{max}}{\triangle x^*} + \frac{|v^*|_{max}}{\triangle y^*} + \frac{|w^*|_{max}}{\triangle z^*}$$

2. External Force Term

- Dimensional Form

$$F_c = \sqrt{\frac{|g|}{\triangle x}}$$

- Dimensionless Form

$$F_c^* = \sqrt{\frac{\frac{R}{V_0^2}|g|}{\triangle x}}$$

3. Surface Tension Term

- Dimensional Form

$$S_c = \sqrt{\frac{\sigma|\kappa|}{\min(\rho_1, \rho_2)(\min(\triangle x, \triangle y, \triangle z))^2}}$$

- Dimensionless Form

$$S_c^* = \sqrt{\frac{\frac{1}{We}|\kappa^*|}{\min(\eta, 1)(\min(\triangle x, \triangle y, \triangle z))^2}}$$

From the semi-implicit viscous solver, the CFL condition for the viscous term

$$V_c = \max(\frac{\mu_1}{\rho_1}, \frac{\mu_2}{\rho_2})(\frac{2}{(\triangle x)^2 + (\triangle y)^2 + (\triangle z)^2})$$

is released.
In our simulation, a CFL restriction of $\frac{1}{2}$ is used. Hence,

$$\triangle t^*(\frac{A_c^* + \sqrt{(A_c^*)^2 + 4(F_c^*)^2 + 4(S_c^*)^2}}{2}) \leq \frac{1}{2}$$

Remark Note that we limit upper bound of curvature with

$$|\kappa| \leq \frac{1}{min(\triangle x, \triangle y, \triangle z)}$$

so that prevent large curvature near the principal axis from contributing erroneous large surface tension forces. This ensures that CFL restriction on surface tension term is $O(h^{3/2})$ in uniform grid which is similar as in [16].

Chapter 4

Numerical experiments

4.1 Two-Phase Core-Annular Flow in Crude Oil Transportation

The transportation of heavy crude oil through a pipe is a naturally important problem in oil industry. Among several efficient modes of transportation, lubricating a pipe with a low-viscosity fluid, such as water, is a well-used technique. The primary idea behind lubricating pipes is the property of low-viscosity fluid to encapsulate high-viscosity fluid. This type of flow is designated as core-annular flow, which features one fluid at the core and another fluid in the annulus.

Many attempts have experimented on [17] and simulated core-annular flow [24, 25]. However, in many cases, a vertical or parallel pipe without gravity is used because it is difficult to analyze parallel pipes owing to the effect of gravity. Additionally, in previous simulations, many researchers assumed that the given flow was axisymmetric in a stable regime, which is not true for a parallel pipe with gravity. Recently, one study [15] attempted to simulate a horizontal core-annular flow by using the VOF method [24], but this simulation examined a 2D cross-sectional plane and a 3D simulation of a specific moment.

The purpose of this study is to perform a full 3D simulation on core-annular flow in a horizontal pipe while considering the gravity effect. To describe

the interface between oil and water, we selected a level set method [6]. In many simulations on incompressible fluid [8, 21], level set formulation is well-suited for visualizing the interface. Because oil and water have very different viscosities, we primarily attempted to perform the simulation based on the boundary capturing method [23, 21]. This is viewed as an extension of Kang's work [22] in terms of using a level set method. However, in 3D simulation, axisymmetric coordinates must be transformed into regular grid coordinates with irregular boundary domain for a cylindrical pipe, which complicates the simulation.
This example provides numerical results on core-annular flow in a parallel pipe in full 3D domain.

4.1.1 Parameters for Equations

In this example, oil is designated as fluid 1 and water as fluid 2. The pipe radius is represented as R_2 and the interface position is $r = R_1$. Regarding the interfacial tension σ, $P_2 - P_1 = \sigma/R_1$. Note that in core annular flow, the pressure in governing equation must be $P = -\mathbf{f}\cdot\mathbf{x} + p$ for the driven pressure gradient vector $\mathbf{f}$ and the axis coordinate vector $\mathbf{x}$. Constant pressure gradient in the axial direction is imposed: $dP/dx = -f$. This constant gradient is viewed as the external force. The four primary dimensionless parameters are given as follows:

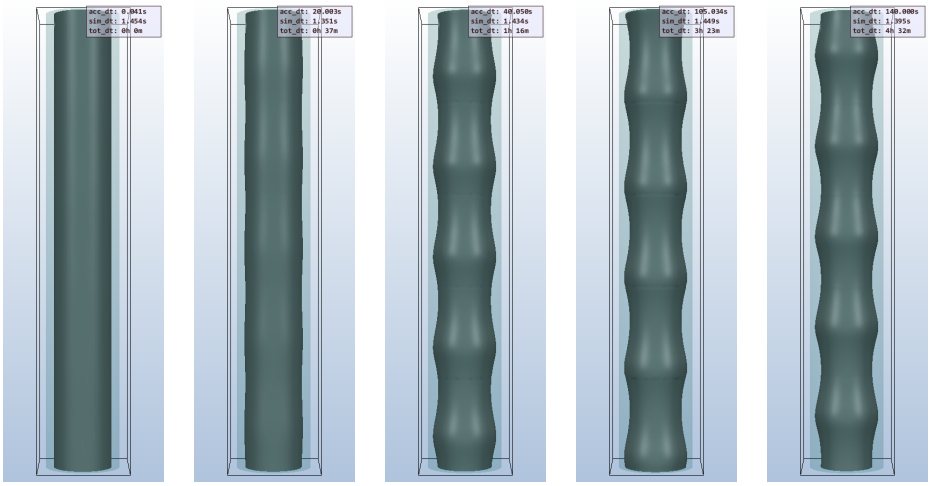
$$m = \mu_2/\mu_1, \quad a = R_2/R_1, \quad \zeta = \rho_2/\rho_1, \quad K = (f + \rho_1 g)/(f + \rho_2 g),$$

where K measures the ratio of driving forces in the core and annulus. The centerline velocity is chosen to be

$$V_0(0) = (f + \rho_2 g)\frac{R_1^2}{4\mu_2}A, \quad A = mK + a^2 - 1 + 2(K-1)\log a$$

The dimensionless base velocity field is $V(r)$, where

$$V(r) = \begin{cases} \frac{1}{A}[a^2 - r^2 - 2(K-1)\log(r/a)], & 1 \le r \le a, \\ 1 - \frac{1}{A}(mr^2K), & r < 1. \end{cases}$$



(a) $t = 0$ (b) $t = 20$ (c) $t = 40$ (d) $t = 105$ (e) $t = 140$

Figure 4.1: Up-flow with $Re_1 = 0.94938$, $a = 1.28$, $m = 0.00166$, $\eta = 1.1$, $J = 0.07961$, $K =$ -0.4552, $V_0 = 10.034\ cm\ s^{-1}$, $\alpha = 2.0$, $C = 0.001$.

where r is the distance from the axis of the axial direction to the interface. The velocity is imposed only for the axial direction.
An interfacial tension parameter is $J = \sigma R_1 \rho_1 / \mu_1^2$. Reynolds numbers, Re_i, are defined by $Re_i = \rho_i V_0(0) R_1 / \mu_i, i = 1, 2$, where $Re_1 / Re_2 = m/\zeta$.

4.1.2 Upflow Case for Validation

We examine an upflow example for validation of our non-axisymmetric based simulation.

Example 1

We simulate the case with $[Q_w, Q_o] = [200, 429] cm^3 min^{-1}$. We set $a = 1.28$ according to equation (3.4) in a previous study [24], with experimental hold-up ratio $h = 1.39$. The experiments are performed using a pipe of radius $R_2 = 0.47625$, with undisturbed interface, $R_1 = R_2/a$. By fixing a and V_o,

where $V_o = Q_o/(\pi R_2^2) = 10.034cm\ s^{-1}$, we can use the other parameters from equation (18.15) in Li and Renardy [28] as follows:

$$Re_1 = 0.9498, \quad m = 0.00166, \quad \eta = 1.1, \quad J = 0.07961, \quad K = -0.4552$$

$$C = 0.001, \quad \alpha = 2.0, \quad V_0^* = 16.9531$$

In dimensionless form, the perturbed amplitude will be $C^* = C/R_1$. This simulation is performed on a $51 \times 51 \times 51$ mesh over one spatial period on domain $[-1.57, 0, -1.57] \times [1.57, 3.14, 1.57]$. Our results for this case are shown in Fig. 4.1.

Example 2

In this example, we establish an amount of water to be relatively large compared with that in example 1, $a = 1.61$. Corresponding parameters for this simulation are as follows:

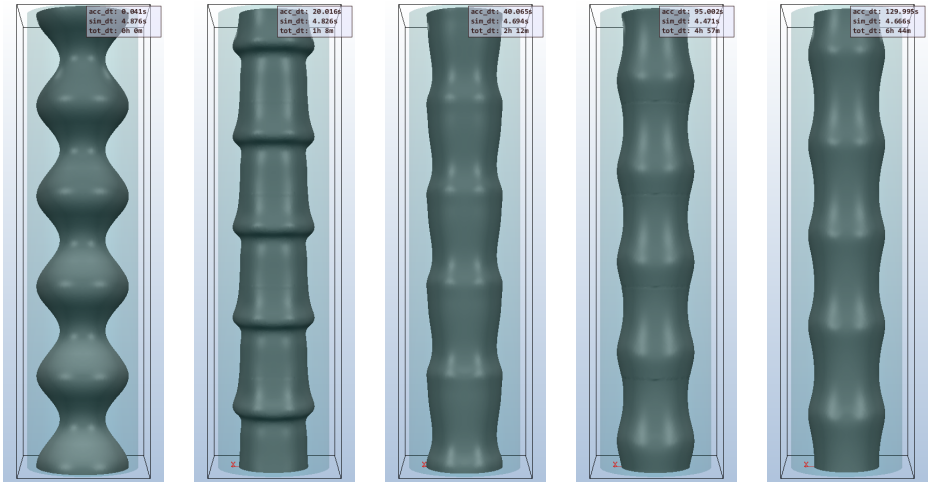
$$Re_1 = 3.73754, \quad m = 0.00166, \quad \eta = 1.1, \quad J = 0.063354, \quad K = -2.030303$$

$$C = 0.1, \quad \alpha = 2.4, \quad V_0^* = 83.91$$

This simulation is performed on a $71 \times 51 \times 71$ mesh over one spatial period on domain $[-1.8326, 0, -1.8326] \times [1.8326, 2.618, 1.8326]$. The results are shown in Figs. 4.2. Including the asymmetric aspect of the crest, these results agree with those in previous studies [24, 22].

4.1.3 Horizontal Flow with Gravity Effect

Now, we discuss the cases of horizontal flow in a pipe. In G.Sotgia *et al.* [27], they performed experiments based on their test facility with 10-meter long pipeline in Fig. 4.3. This is quite long compared with its cross-sectional radius which give us a limitation of computation. Also, It is known that there exist parameters which make fluid motion to be periodic for core-annular flow by experiments in [27]. In this paper, we are focusing on periodic regime for comparing with recent results in [15] by choosing parameters as the same with the one of periodic regime case. We assume that the flow moves from right to left by negative pressure gradient.



(a) $t = 0$ (b) $t = 20$ (c) $t = 40$ (d) $t = 95$ (e) $t = 130$

Figure 4.2: Up-flow with $Re_1 = 3.73754$, $a = 1.61$, $m = 0.00166$, $\eta = 1.1$, $J = 0.063354$, $K =$ -2.030303, $V_0 = 83.91\ cm\ s^{-1}$, $\alpha = 2.4$, $C = 0.1$.

Example 3

In this example, we add more water than in the vertical case, and let the core-centerline be sufficiently fast for choosing similar parameter with [27, 15]. Corresponding parameters are as follows:

$$Re_1 = 3.73754, \quad m = 0.00166, \quad \eta = 1.1, \quad J = 0.063354, \quad K = -2.030303$$

$$C = 0.01, \quad \alpha = 2.4, \quad V_0^* = 83.91$$

Except for the amplitude of initial wave, parameters are as same with as those in example 2. Because water is relatively heavier than oil, a large amount levitates the oil. We can verify this effect from Fig. 4.5 and we can see the flow pattern quite agrees on experimental case, Fig 4.4.

Example 4

We choose similar parameters to those in example 1, except K.

$$Re_1 = 0.9498, \quad m = 0.00166, \quad \eta = 1.1, \quad J = 0.07961, \quad K = -0.9993$$

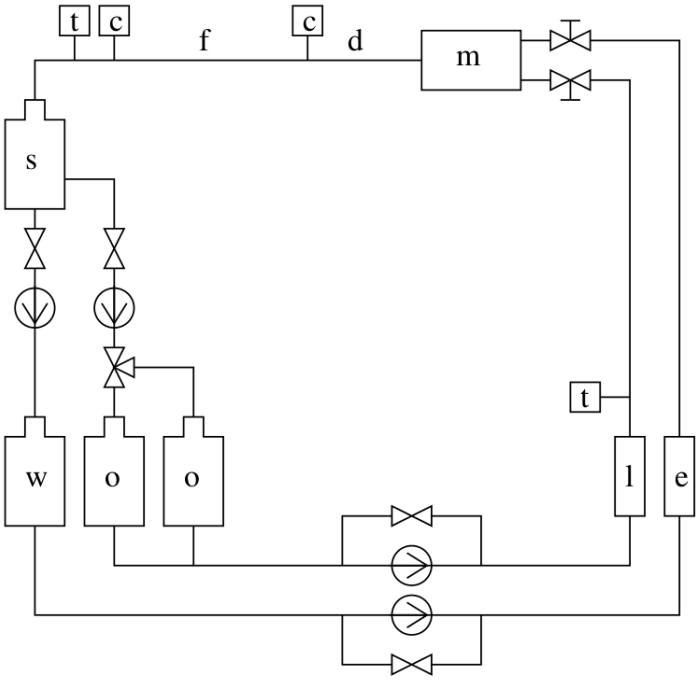


Figure 4.3: Schematic representation of the oil water loop, (o) oil Reservoir, (w) water reservoir, (l) laminar flowmeter, (e) electromagnetic flow meter, (t) thermocouple, (m) two-phase mixer, (d) flow development region, (f) fully developed flow, (c) capacitance pressure transducer, (s) separator tank.

$$C = 0.001, \quad \alpha = 2.0, \quad V_0^* = 53.4918$$

In this case, the density difference between two fluids is relatively large (ρ_1 - $\rho_2 = 0.090$), otherwise the centerline velocity will be relatively low. Hence, we can expect that the oil will first move downward due to the gravity. However, a few seconds later, as the total velocity increases, oil begins to levitate. Additionally, we can verify that the total amount of water is increased as time passes. The results are shown in Fig. 4.6.

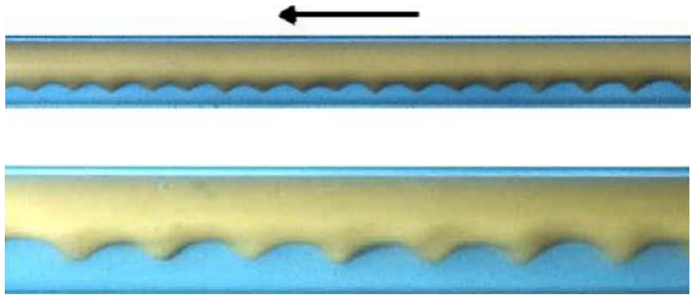
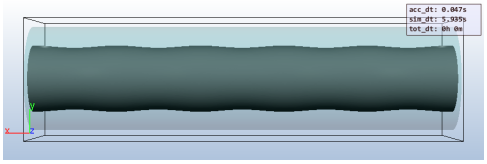
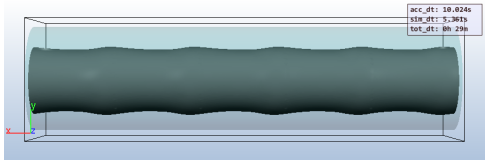


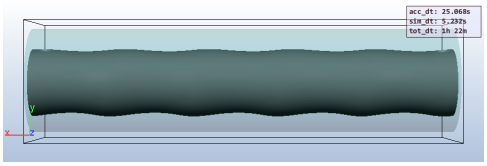
Figure 4.4: G.Sotgia, P. Tartarini and E. Stalio [27]. The periodicity of wave is varying depending on the oil and water velocity



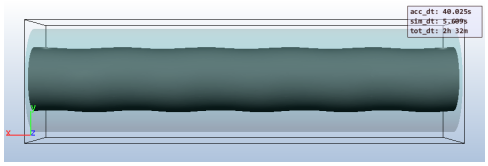
(a) $t = 0$



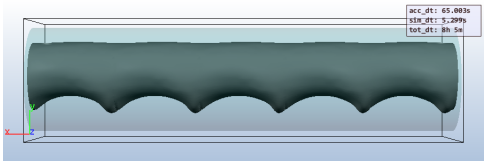
(b) $t = 10$



(c) $t = 25$



(d) $t = 40$

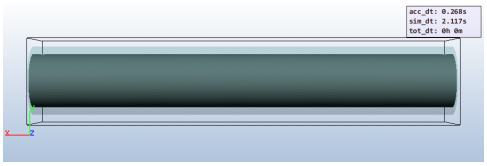


(e) $t = 65$

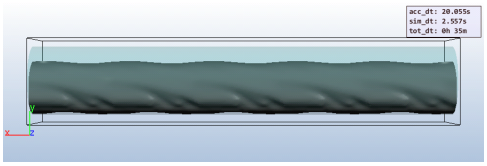
Figure 4.5: Horizontal flow with $Re_1 = 3.73754$, $a = 1.61$, $m = 0.00166$, $\eta = 1.1$, $J = 0.063354$, $K =$ -2.030303, $V_0 = 83.91\ cm\ s^{-1}$, $\alpha = 2.4$, $C = 0.01$

Example 5: Effect of Centerline Velocity

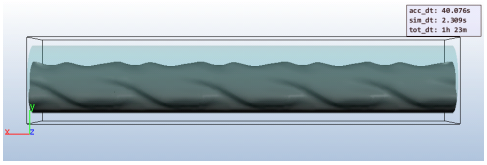
To verify the effect of changing the centerline velocity, we change K from the value in example 4 into $K = -0.4552$ and $K = -2.030303$. We can con-



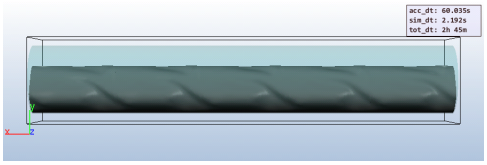
(a) $t = 0$



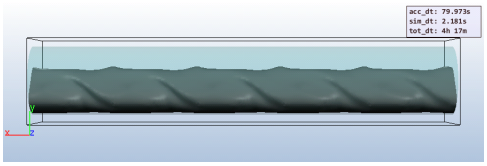
(b) $t = 20$



(c) $t = 40$



(d) $t = 60$



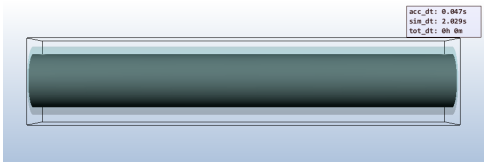
(e) $t = 80$

Figure 4.6: Horizontal flow with $Re_1 = 0.94938$, $a = 1.28$, $m = 0.00166$, $\eta = 1.1$, $J = 0.07961$, $K =$ -0.9993, $V_0 = 53.4918\ cm\ s^{-1}$, $\alpha = 2$, $C = 0.001$.

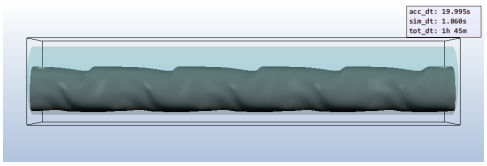
clude that the centerline velocity of the core flow affects the wavy crest and thickness of the core flow and the height of levitation. Because the amount of water is relatively low compared to example 3, the height of levitation is low. Additionally, we can verify that when core flow moves around the pipe, a wavy interface occurs because of the buoyancy force from the water near the pipe. Figs. 4.7, 4.8 give results.

Example 6: Effect of Finite Amplitude on Core Flow

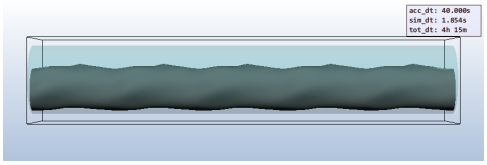
In this example, we vary the finite amplitude from that in example 4 to $C = 0.01$ and $C = 0.05$. Figs. 4.9 and 4.10 show each result. The perturbed amplitude for the initial pipe can affect the pressure around the pipe wall. The pipe with relatively large amplitude tends to be smoother than the other,



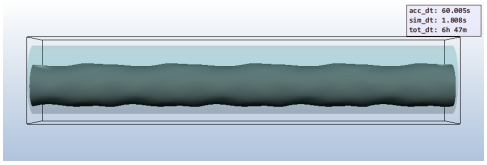
(a) $t = 0$



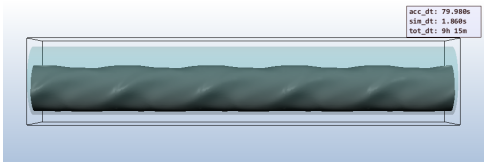
(b) $t = 20$



(c) $t = 40$



(d) $t = 60$



(e) $t = 80$

Figure 4.7: Horizontal flow with $Re_1 = 0.94938$, $a = 1.28$, $m = 0.00166$, $\eta = 1.1$, $J = 0.07961$, K = -0.4552, $V_0 = 10.034\ cm\ s^{-1}$, $\alpha = 2$, $C = 0.001$.

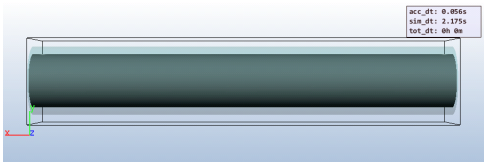
and carries more of the core flow due to the initial pressure profile.

Example 7: Effect of Radius Ratio

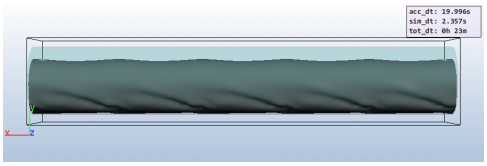
Next, we verify the effect of changing the amount of water by controlling a. Let us change a to $a = 1.61$. The amount of water plays an important role in the levitation of the core flow. As the amount of fluid in the annulus grows, the buoyancy force from the annulus strengthens. This makes the core fluid levitate from the pipe wall. Flows and pressure profiles can be verified in Fig. 4.11.

Example 8: Effect of Reynolds Number

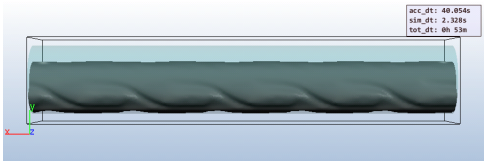
Finally, we investigate the effects of several different Reynolds numbers, Re_1, by varying the Reynolds number in example 3 to $Re_1 = 1.5$ and $Re_1 = 2.5$. As



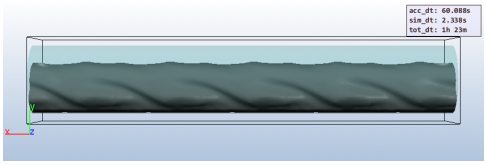
(a) $t = 0$



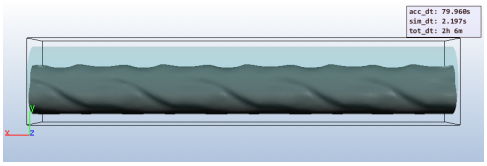
(b) $t = 20$



(c) $t = 40$



(d) $t = 60$



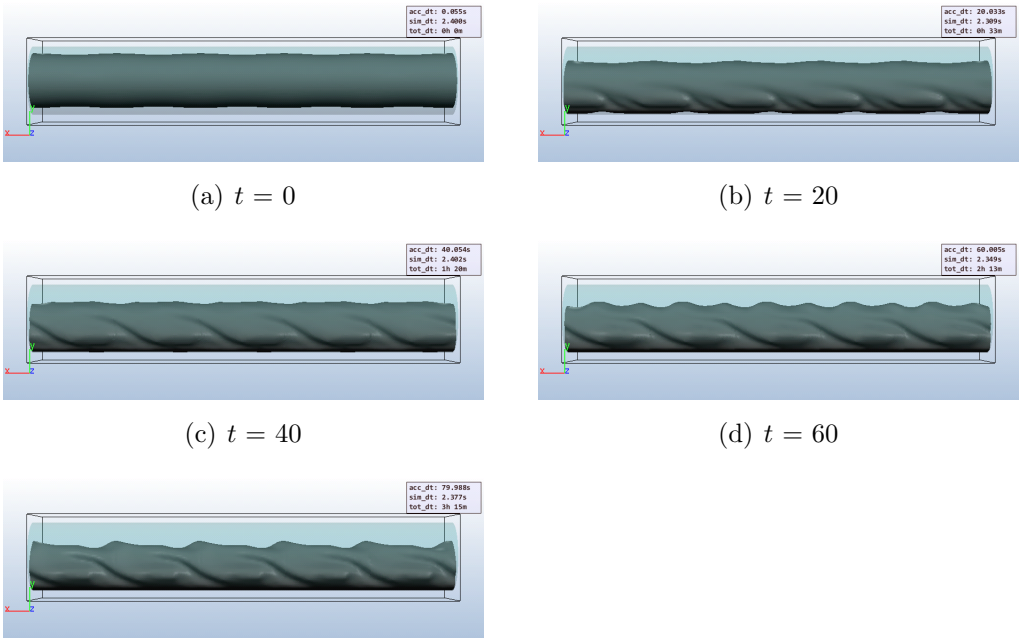
(e) $t = 80$

Figure 4.8: Horizontal flow with $Re_1 = 0.94938$, $a = 1.28$, $m = 0.00166$, $\eta = 1.1$, $J = 0.07961$, $K =$ -2.030303, $V_0 = 83.91\ cm\ s^{-1}$, $\alpha = 2$, $C = 0.001$.

the Reynolds number grows, the viscosity effect of the core flow decreases. Hence, we can expect that the core flow will levitate to a relatively high height when the Reynolds number is small. Figs. 4.12 and 4.13 show that this expectation is correct.

4.1.4 Time for Computing

For simulations, we use a computer with a 3.5 Ghz Intel(R) i7-2700K processor. Simulation is coded by using Visual C++, OpenGL as a graphical tool, and Boost as a multithreading tool. We use eight cores for multithreading. Total and simulation times for one step of each simulation are represented on the right edge of each figure.

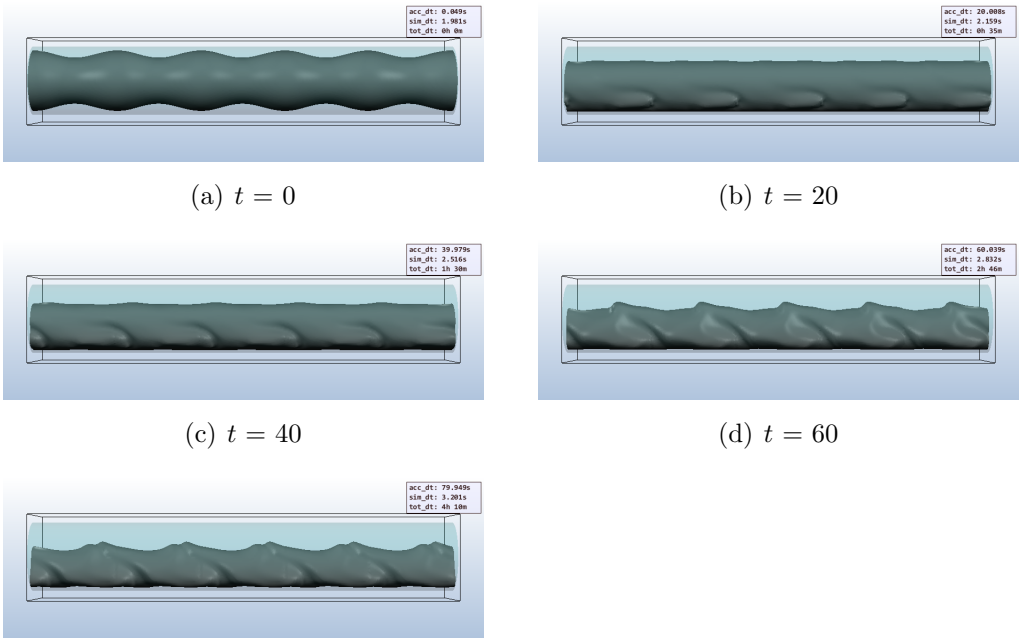


(a) $t = 0$ (b) $t = 20$

(c) $t = 40$ (d) $t = 60$

(e) $t = 80$

Figure 4.9: Horizontal flow with $Re_1 = 0.94938$, $a = 1.28$, $m = 0.00166$, $\eta = 1.1$, $J = 0.07961$, $K =$ -0.9993, $V_0 = 53.4918\ cm\ s^{-1}$, $\alpha = 2$, $C = 0.01$.

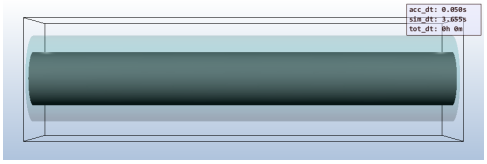


(a) $t = 0$ (b) $t = 20$

(c) $t = 40$ (d) $t = 60$

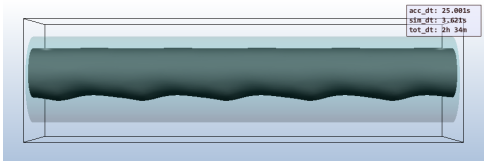
(e) $t = 80$

Figure 4.10: Horizontal flow with $Re_1 = 0.94938$, $a = 1.28$, $m = 0.00166$, $\eta = 1.1$, $J = 0.07961$, $K =$ -0.9993, $V_0 = 53.4918\ cm\ s^{-1}$, $\alpha = 2$, $C = 0.05$.

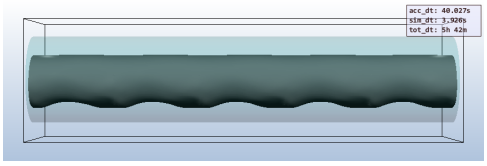


(a) $t = 0$

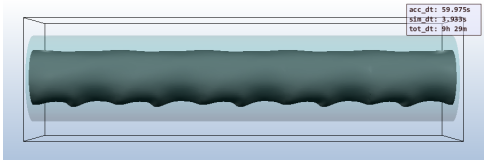
(b) $t = 10$



(c) $t = 25$

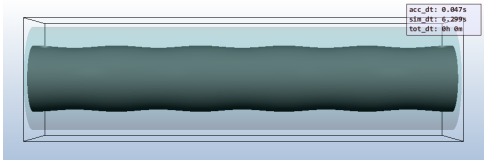


(d) $t = 40$

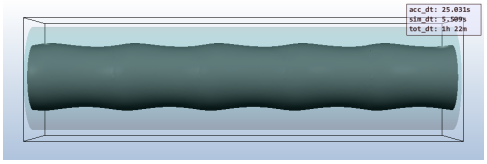


(e) $t = 60$

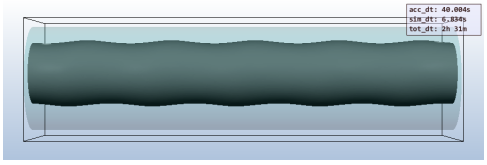
Figure 4.11: Horizontal flow with $Re_1 = 0.94938$, $a = 1.61$, $m = 0.00166$, $\eta = 1.1$, $J = 0.07961$, $K =$ -0.9993, $V_0 = 53.4918\ cm\ s^{-1}$, $\alpha = 2$, $C = 0.01$.



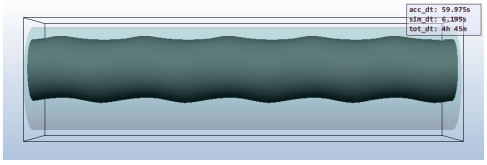
(a) $t = 0$



(b) $t = 25$

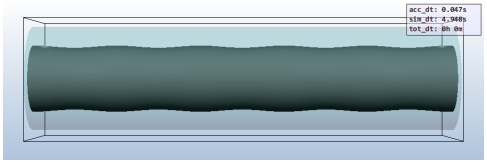


(c) $t = 40$

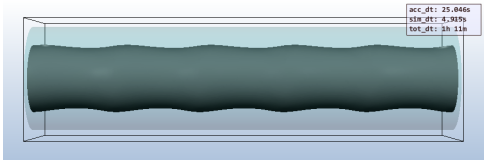


(d) $t = 60$

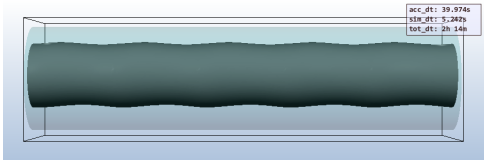
Figure 4.12: Horizontal flow with $Re_1 = 1.5$, $a = 1.61$, $m = 0.00166$, $\eta = 1.1$, $J = 0.063354$, $K =$ -2.030303, $V_0 = 83.91\ cm\ s^{-1}$, $\alpha = 2.4$, $C = 0.01$.



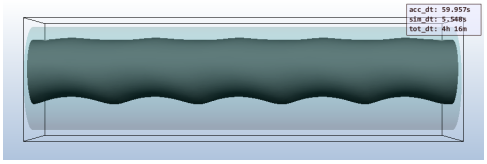
(a) $t = 0$



(b) $t = 25$



(c) $t = 40$



(d) $t = 60$

Figure 4.13: Horizontal flow with $Re_1 = 2.5$, $a = 1.61$, $m = 0.00166$, $\eta = 1.1$, $J = 0.063354$, $K =$ -2.030303, $V_0 = 83.91\ cm\ s^{-1}$, $\alpha = 2.4$, $C = 0.01$.

Chapter 5

Conclusion

We present simulations of core-annular flow with a large viscosity ratio in a horizontal pipe equipped with a level set method. Comparing this result with recent work [15], we provide a full 3D description of core-annular flows in parallel pipes with several examples. By selecting different parameters, we can ensure that the core fluid flows downward due to the relatively large density difference. Additionally, in our simulation, the oil-water interface is described accurately by means of the level set function. To the best to our knowledge, full 3D simulations on horizontal pipes have not been conducted before. Such simulations can anticipate real phenomena in transporting viscous liquid such as crude oil.

Because analysis on horizontal pipes is not fully established, we have no choice but to select parameters from the vertical case. In reality, the assumption of periodicity will not be accurate, which is a limitation of our simulation. In future work, we will consider finding a proper parameter for efficient transportation based on the foregoing analysis. Additionally, we will extend the given simulation to long pipes without periodicity assumption, and differently located pipes (such as inclined or curved).

Bibliography

[1] Randall J. LeVeque, *Numerical Methods for Conservation Laws*, vol2, Birkhauser Verlag, 1992

[2] Lluyd N. Trefethen, David Bau, *Numerical Linear Algebra*,Siam, 1992

[3] Chorin, A.J. *Numerical solution of the Navier-Stokes equations*, Math. Comp. **22**, 745 (1968).

[4] Roger Peyret, Thomas D. Taylor, *Computational Methods for Fluid Flow*, Springer-Verlag, 1982

[5] Eleuterio F. Toro, *Riemann Solvers and Numerical Methods for Fluid Dynamiacs*, third edition, Springer, 2009

[6] S. Osher, J. A. Setian, *Fronts Propagating with Curvature Dependent Speed: Algorithms Based on Hamilton-Jacobi Formulations*, J. Comput. Phys. **79**, 12 49 (1988).

[7] S. Osher, and Shu, C.W. *High-order essentially non-oscillatory schemes for Hamilton-Jacobi equation*, SINUM. **28**, 907 922 (1991).

[8] Mark Sussman, Peter Smereka, S. Osher, *A Level Set Approach for Computing Solutions to Incompressible Two-Phase Flow*, J. Comput. Phys. **114)**, 146 159 (1994).

[9] Lawrence C. Evans, *Partial Differential Equations*, American Mathematical Society, 1997

[10] L.D.Landau, E.M.Lifshitz, *Fluid Mechanics*, second edition, Pergamon Press, 1987

[11] S.O. Unverdi, G. Tryggvason, *A front-tracking method for viscous, incompressible, multi-fluid flows* J. Comput. Phys. **100**, 25 (1992).

[12] C.R Anderson, *A vortex method for flows with slight density variations*, J. Comput. Phys. **61**, 3 (1985).

[13] G.R. Baker, D.W. Moore, *The rise and distortion of a two-dimensional gas bubble in an inviscid liquid*, Phys. Fluid A **1**, (1989).

[14] C.W. Hirt, B.D. Nichols, *Volume of fluid method for the dynamics of free boundaries*, J. Comput. Phys. **39**,201, (1981).

[15] Ooms, G., Pourquie, M.J.B.M., Beerens, J.C. *On the levitation force in horizontal core-annular flow with a large viscosity ratio and small density ratio* Phys. Fluids **25**, 032102 (2013).

[16] Brackbill, J.U., Kothe, D.B., Zemach, C. *A continuum method for modeling surface tension*, J. Comput. Phys. **100**, 335-354 (1992).

[17] Randall J. LeVeque, *Finite Differnce Methods for Ordinary and Partial Differential Equations*, Siam, 2007

[18] S. Osher, Ronald Fedikew *Level Set Methods and Dynamic Implicit Surfaces*, Springers, 2003

[19] C.W. Shu, S. Osher, *Efficient Implementation of essentially non-oscillatory shock capturing schemes* J. Comput. Phys. **83**,32 ,(1988).

[20] Russo, G., Semereka, P. *A Remark on Computing Distance Functions* J. Comput. Phys. **163**, 51-67 (2000).

[21] Kang, M., Fedkiw, R.P., Liu, X.D. *A boundary condition capturing method for multiphase incompressible flow*, J. Sci. Comput. **15**, 323 360 (2000).

[22] Kang, M., Shim, H., Osher, S. *Level set based simulations of two-phase oil-water flows in pipes*, J. Sci. Comput. **31**, 153 (2006).

[23] Liu, X.D., Fedkiw, R.P., Kang, M. *A boundary condition capturing method for Poisson's equation on irregular domains*, J. Comput. Phys. **160**, 151 178 (2000).

[24] Li, J., Renardy, Y.Y., Renardy, M. *Direct simulation of unsteady axisymmetric core-annular flow with high viscosity ratio*, J. Fluid. Mech. **391**, 123-149 (1998).

[25] Li, J., Renardy, Y.Y., Renardy, M. *A numerical study of periodic disturbances on two-layer Couette flow*, Phys. Fluids. **10(12)**, 3056 3071 (1998).

[26] J.W. Purvis, J.E. Burkhalter. *Prediction of critical mach number for store configurations* AIAA J. **17**, 1170-1177 (1979).

[27] G. Sotgia, P. Tartarini, and E. Stalio. *Experimental analysis of flow regimes and pressure drop reduction in oil-water mixtures* Int. J. Multiphase Flow **34**, 1161 (2008).

[28] Joseph, D.D., Renardy, Y.Y. *Fundamentals of Two-Fluid Dynamics, Part II, Lubricated Transport, Drops and Miscible Liquids*, Springer-Verlag, New York (1993).

[29] Jiang, G.S., Peng, D. *Weighted ENO schemes for Hamilton-Jacobi equations*, SIAM J. Sci. Comput. **21**, 2126 2143 (2000).

[30] JR Shewchuk, *An introduction to the conjugate gradient method without the agonizing pain*, unucauca.edu.co. (1994)

[31] D.Xiu., G.E.Karniadakis. *A semi-Lagrangian high-order method for Navier-Stokes equations*, J. Comput. Phys. **172(2)**, 658 684(2001).

[32] Randall J. LeVeque, Zhilin Li. *The Immersed Interface Method for Elliptic Equations with Discontinuous Coefficients and Singular Sources*, SIAM J. Num. Anal. **31(4)**, 1019 1044(1994)

[33] A. Harten, B. Engquist, S. Osher, S. Chakravarthy,. *Uniformly High-Order Accurate Essentially Non-Oscillatory Schemes III*, J. Comput. Phys. **71**, , 231.303 (1987)

국문초록

본 논문에서는 레벨셋 방법을 이용한 다층 유체를 시뮬레이션하는 수치적 방법을 다룬다. 다층 유체의 운동은 비압축성 나비에-스톡스 방정식을 통해서 표현할 수 있다. 우선, 유체의 지배방정식의 해를 근사적으로 풀어내는 수치적 방법들에 대해서 살펴본다. 그리고, 유체의 경계면을 표현하는 레벨셋 방법을 소개하고 나비에-스톡스 방정식과의 결합을 소개한다. 마지막으로, 평행한 파이프에서의 환상 핵심 순환 유체의 수치적 시뮬레이션 결과를 보여준다.

주요어휘: 다층 유체, 레벨셋 방법, 수치적 시뮬레이션, 나비에-스톡스 방정식, 환상 핵심 순환
학번: 2011-30900